SISTEM PENCARIAN JURNAL ILMIAH *CROSS LANGUAGE* DENGAN METODE *VECTOR SPACE MODEL* (VSM)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Indah Mutia Ayudita
NIM: 145150201111076

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK IFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

SISTEM PENCARIAN JURNAL ILMIAH *CROSS LANGUAGE* DENGAN METODE *VECTOR SPACE MODEL* (VSM)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Indah Mutia Ayudita
NIM: 145150201111076

Skripsi ini telah diuji dan dinyatakan lulus pada
1 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Indriati, S.T., M.Kom.	Putra Pandu Adikara, S.Kom., M.Kom.
NIP. 19831013 2015 04 2 002	NIP. 19850725 200812 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 18 Juli 2018

Indah Mutia Ayudita
NIM: 145150201111076

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Sistem Pencarian Jurnal Ilmiah *Cross Language* Dengan Metode *Vector Space Model* (Vsm)" ini dapat terselesaikan. Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Ibu Indriati, S.T., M.Kom. dan Bapak Putra Pandu Adikara, S.Kom., M.Kom. selaku Pembimbing skripsi yang telah membimbing, memberi saran, dan mengarahkan penulis selama penyusunan laporan skripsi,
2. Bapak Agus Wahyu Widodo, S.T, M.Cs. selaku Ketua Program Studi Teknik Informatika,
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku Ketua Jurusan Teknik Informatika,
4. Bapak Marji, Drs., M.T. selaku dosen Penasihat Akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi,
5. Dosen Fakultas Ilmu Komputer yang telah mendidik dan memberikan ilmu selama penulis menempuh pendidikan di Fakultas ini,
6. Orang tua dan saudara penulis yang selalu mendukung penulis baik secara moril dan materiil sedari awal menempuh pendidikan hingga penulis dapat menyelesaikan skripsi ini,
7. Seluruh keluarga besar kedua orang tua penulis yang selalu memberikan semangat,
8. Cincau: Alqis, Kiki, Nisa dan Ajeng yang sudah bersama dari 2014 sampai sekarang yang selalu bersabar, menemani, membantu, memberi semangat dan dukungan sampai saat ini,
9. Udik: Feni, Meli, Wahyu, Ira, Karin, dan Stef, yang dari dulu menjadi wadah bertukar cerita dan berbagi pengalaman dan selalu siap membantu,
10. Teman-teman yang belum bisa ditemui secara langsung, namun tetap memberikan dukungan dan semangat serta bantuannya: Laras, Allyza, Nay, dan Runni,
11. Teman-teman kelas F Informatika atas kenangannya selama bersama-sama dulu
12. Teman-teman BPMIF dan angkatan 2014 yang membantu proses pembuatan skripsi ini, terus memberikan dukungan dan semangat
13. Seluruh pihak yang telah membantu kelancaran penulisan tugas akhir yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa penulisan skripsi ini jauh dari kata sempurna. Maka untuk itu penulis mengharapkan saran, kritik, serta masukan dari semua pihak, demi kesempurnaan dan perbaikan skripsi ini. Akhir kata semoga skripsi ini dapat memberikan manfaat bagi semua pihak.

Malang, 18 Juli 2018

Penulis
indahmutiayudt@gmail.com

ABSTRAK

Indah Mutia Ayudita, Sistem Pencarian Jurnal Ilmiah *Cross Language* dengan Metode *Vector Space Model* (*VSM*)

Pembimbing: Indriati, S.T., M.Kom. dan Putra Pandu Adikara, S.Kom., M.Kom.

Jurnal ilmiah adalah majalah publikasi yang memuat karya tulis ilmiah yang secara nyata mengandung data dan informasi yang ditulis sesuai dengan aturan penulisan ilmiah serta diterbitkan secara berkala. Jurnal ilmiah banyak dimanfaatkan sebagai acuan untuk membuat penelitian baru atau melanjutkan penelitian yang sudah ada. Jurnal ilmiah saat ini mudah ditemui dalam bentuk digital dan tersedia pada perpustakaan *online* seperti Science Direct dan IEEE, namun pencarian yang dilakukan masih terbatas pada *Monolingual Information Retrieval*, yang mana hasil pencarian memiliki bahasa yang sama dengan bahasa *query* masukan. Namun, dokumen yang relevan tidak terbatas hanya pada bahasa yang sama dengan *query*. Penelitian ini dilakukan untuk melihat hasil implementasi *Cross Language Information Retrieval* yang mampu melakukan pencarian dengan masukan satu bahasa dan menghasilkan dokumen dalam dua bahasa. Hasilnya, 8 dari 10 query mengalami kenaikan *precision* sampai 74,5% dan *recall* sampai 41,5%. Secara umum, akurasi sistem dalam menampilkan dokumen yang relevan rata-rata sebesar 84%.

Kata kunci: *cross language information retrieval, vector space model, jurnal ilmiah.*

ABSTRACT

Indah Mutia Ayudita, ***Cross Language Search Engine for Scientific Journals Using Vector Space Model (VSM)***

Supervisors: Indriati, S.T., M.Kom. ***and*** **Putra Pandu Adikara, S.Kom., M.Kom.**

Scientific journals are periodical publications that contain scientific papers with data and information written in accordance with the rules of scientific writing. Scientific journals used widely as a reference to make a new research or continue the previous research. As the usage is growing, scientific journals also easier to find digitally and available in a digital library such as Science Direct and IEEE, but the searching process is still limited to Monolingual Information Retrieval, in which the search results have the same language as the query inputted, even though the relevant documents also available in other languages. This research is done to observe the result of implementing Cross Language Information Retrieval that can do the searching process in one language for the input and retrieved document in two languages. The final result is 8 out of 10 queries have a higher precision up to 74,5% and recall up to 41,5%. Generally, system can retrieved the relevant documents in average for 84%.

Keywords: cross language information retrieval, vector space model, jurnal ilmiah.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar belakang

Jurnal ilmiah adalah majalah publikasi yang memuat karya tulis ilmiah (KTI) yang secara nyata mengandung data dan informasi yang mengajukan iptek dan ditulis sesuai dengan kaidah-kaidah penulisan ilmiah serta diterbitkan secara berkala. Saat ini, jurnal ilmiah semakin mudah ditemui karena regulasi dari Direktorat Jendral Pendidikan Tinggi (Dikti) yang menghimbau mahasiswa ditingkat S1, S2, maupun S3 untuk membuat karya tulis berupa jurnal ilmiah. Jurnal ilmiah juga dibutuhkan sebagai acuan untuk membuat penelitian baru atau melanjutkan penelitian tersebut agar hasilnya lebih maksimal.

Ada beberapa cara yang dapat dilakukan untuk mendapatkan referensi dalam penyusunan laporan penelitian, antara lain dengan mencari jurnal dan buku dalam bentuk tercetak di perpustakaan, atau dengan memanfaatkan dokumen-dokumen yang ada secara digital. Beberapa perpustakaan jurnal *online* yang sering digunakan antara lain Science Direct dan IEEEXplore untuk jurnal internasional, serta jurnal LIPI untuk jurnal nasional. Mengutip dari penelitian yang dilakukan oleh Ibnu Rusydi, beliau menyatakan bahwa *e-journal* (jurnal dalam bentuk digital) lebih diminati karena dapat menghemat ruang dan waktu, aksesnya yang mudah, mudah dibawa dan didistribusikan sesuai kebutuhan, dan harganya lebih murah (Rusydi, 2014).

Untuk menunjang perkembangan dan distribusi jurnal ilmiah ini, diperlukan suatu sistem untuk menemukan kembali informasi yang dibutuhkan pengguna. Sistem temu kembali informasi digunakan untuk menemukan informasi-informasi yang relevan terhadap kata kunci (*query)* yang dimasukkan pengguna dari sekumpulan dokumen atau korpora. Salah satu aplikasi umum dari sistem temu kembali ini adalah mesin pencarian yang ada pada jaringan internet. Saat ini, kebanyakan mesin pencarian masih bekerja secara *monolingual* atau satu bahasa.

Monolingual Information Retrieval (MLIR) akan menampilkan dokumen sesuai dengan bahasa masukan pengguna. Namun, dengan perkembangan zaman, ada banyak dokumen-dokumen lain dalam berbagai bahasa yang sebenarnya sesuai dengan *query* pencarian pengguna, tetapi tidak dapat ditampilkan sebagai hasil yang relevan. Untuk meningkatkan hasil pencarian dokumen sedang dikembangkan suatu metode yang dapat menghasilkan lebih dari satu bahasa sebagai hasil pencarian. Sistem ini disebut sebagai *Cross Language Information Retrieval (CLIR)* (Zakir, 2015) .

Pengembangan sistem lintas bahasa atau *cross language* dilakukan dengan harapan hasil temu kembali menjadi lebih relevan dengan kata kunci pencarian. Selain itu, *cross language* juga diperlukan untuk memecah penghalang penggunaan bahasa untuk memenuhi kebutuhan akses informasi yang multibahasa, dalam penelitian ini digunakan studi kasus untuk dokumen-dokumen jurnal ilmiah dibidang ilmu komputer dalam bahasa Indonesia dan Inggris.

Penerapan CLIR dapat diimplementasikan dalam beberapa model, antara lain, *probabilistic model*, *set-theoretic model*, dan *algebratic model*. *Vector space model (VSM)* sebagai salah satu contoh dari *algebratic model* adalah model yang paling sederhana, namun efektif dalam pencarian kata. *Vector space model* mengukur kemiripan vektor *query* dan vektor dokumen dan mengurutkannya berdasar kemiripan tertinggi (Bari & Saputra, 2011, disitasi oleh Zakir, 2015).

Dalam penelitiannya tentang perbandingan antar model *information retrieval* untuk bahasa Malayalam, Arjun dan Sindhu menyatakan bahwa hasil temu kembali dengan metode v*ector space model* menghasilkan nilai yang lebih baik dibandingkan model lain, seperti *boolean model,* dan *probabilistic model*, terutama untuk dokumen berukuran besar karena selain mengukur kemiripan juga mengurutkan dokumen sesuai dengan nilai kemiripannya (Babu & Sindhu, 2014).

Sharma dan Mittal melakukan penelitian untuk sistem *cross language information retrieval* dengan membandingkan metode *term frequency* dengan metode *probabilistic lexicon*. Hasilnya metode *term frequency* (*TFM*) lebih baik dari pada metode *probabilistic lexicon (PLM)* karena *PLM* membutuhkan waktu yang lebih lama dan membutuhkan korpus yang lebih besar dibandingkan dengan *TFM* (Sharma & Mittal, 2016).

Dengan dibuatnya sistem ini, diharapkan dapat memangkas waktu pencarian jurnal ilmiah yang relevan dengan *query* masukan pengguna menggunakan *vector space model*, juga memecahkan batas bahasa khususnya Bahasa Indonesia dan Bahasa Inggris dalam proses pencarian jurnal ilmiah.

1.2 Rumusan masalah

1. Bagaimana akurasi metode *vector space model* dalam menemukan dokumen yang relevan sesuai dengan *query* yang diminta pengguna?
2. Bagaimana pengaruh penggunaan *cross-language* terhadap *precision* dan *recall* hasil temu kembali?

1.3 Tujuan

1. Menemukan akurasi dari metode yang digunakan dalam pencarian dokumen yang revelan sesuai dengan *query* dari *user*.
2. Menemukan pengaruh penggunaan *cross-language* dalam mengimplementasikan sistem temu-kembali terhadap hasil *precision* dan *recall.*

1.4 Manfaat

1. Bagi Penulis
   Dapat mengaplikasikan teori yang telah diperoleh selama perkuliahan secara nyata
2. Bagi Peneliti Selanjutnya
   Dapat menjadi referensi dalam pengembangan sistem lain yang serupa tentang Sistem Temu Kembali Informasi Lintas Bahasa

3. Bagi Pengguna
   Dapat menemukan dokumen jurnal ilmiah berbahasa Inggris dan Indonesia sesuai *query* yang dimasukkan *user*

1.5 Batasan masalah

Penelitian ini digunakan untuk menemukan dokumen jurnal ilmiah dalam Bahasa Inggris dan Bahasa Indonesia dalam satu *query*. Penelitian ini akan dibatasi untuk hal-hal berikut:

1. Dokumen yang digunakan adalah dokumen jurnal ilmiah dalam Bahasa Inggris dan Bahasa Indonesia
2. Dokumen berupa teks dan bagian yang akan digunakan adalah abstrak jurnal ilmiah
3. *Query* yang digunakan adalah teks dan dapat menggunakan Bahasa Inggris atau Bahasa Indonesia

1.6 Sistematika pembahasan

Sistematika pembahasan untuk penulisan penelitian ini adalah sebagai berikut:

BAB I PENDAHULUAN

Menguraikan tentang latar belakang, menemukan dan merumuskan inti permasalahan, menentukan tujuan dan manfaat penelitian, pembatasan masalah penelitian, informasi sistematika penulisan skripsi.

BAB II LANDASAN KEPUSTAKAAN

Membahas tentang konsep dasar dan teori-teori yang berhubungan dengan topik penelitian yang akan dilakukan.

BAB III METODOLOGI PENELITIAN

Bagian ini berisi tentang metode penelitian yang sesuai dengan pendekatan yang digunakan dalam pembuatan sistem.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini berisi tentang perancangan pelaksanaan penelitian dan implementasi penelitian yang dilakukan.

BAB V PEMBAHASAN

Berisi tentang penjelasan dari hasil yang telah diperoleh dalam penelitian.

BAB VI PENUTUP

Berisi tentang kesimpulan yang didapatkan dari penelitian yang telah dilakukan serta saran-saran untuk pengembangan selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Penelitian Terdahulu

Penelitian tentang implementasi metode VSM dalam *information retrieval* telah dilakukan oleh Arjun dan Sindhu (2014). Dalam penelitiannya tentang perbandingan antar model *information retrieval* untuk bahasa Malayalam, Arjun dan Sindhu menyatakan bahwa hasil temu kembali dengan metode v*ector space model* menghasilkan nilai yang lebih baik dibandingkan model lain, terutama untuk dokumen berukuran besar karena selain mengukur kemiripan juga mengurutkan dokumen sesuai dengan nilai kemiripannya (Babu & Sindhu, 2014).

Metode VSM juga digunakan oleh Rozanda, Marsal, dan Iswanti (2013) dalam penelitiannya tentang Sistem Informasi Hadits dengan Teknik Temu Kembali Informasi Model Ruang Vektor. Basis data yang digunakan adalah dokumen hadist berjumlah 536 dokumen dan hasilnya adalah proses pencarian dokumen hadist yang lebih cepat, lebih mudah, dan lebih relevan dengan *query* (Rozanda et al., 2013).

Sari dan Adriani (2014) pernah melakukan penelitian tentang pengurutan dokumen relevan dalam *Cross Language*. Sari dan Adriani menggunakan *Support Vector Machine* (SVM) untuk menentukan relevansi dokumen, dan menggunakan BM25 sebagai metode perankingannya. Penelitian dilakukan pada CLIR berbahasa Inggris dan Indonesia. Hasilnya, kemampuan SVM dalam menemukan dokumen yang relevan sebesar 88.51%, sedangkan akurasi SVM dalam menemukan dokumen non-relevan sebesar 88% (Sari & Adriani, 2014).

Ujjwal, Rastogi, dan Siddhartha (2016) melakukan analisis tentang performa model IR dengan menggunakan parameter empiris yang berbeda dengan menggunakan korpus bebahasa Inggris dan India. Ujjwal, Rastogi, dan Siddharta membandingkan nilai NDCG untuk dokumen top 1, top 5 dan top 10. Hasilnya model Hiemstra memberikan hasil yang paling bagus dari lima model yang diuji yaitu sebesar 82% untuk dokumen top 1 (Ujjwal et al., 2016).

2.2 Mesin Pencarian (*Search Engine)*

Mesin pencarian adalah program komputer yang dirancang untuk melakukan pencarian informasi dari data-data yang tersedia. Hasil pencarian biasanya ditampilkan dalam bentuk daftar yang diurutkan berdasar relevansinya dengan *query* atau kata kunci yang dimasukan oleh pengguna. Mesin ini adalah hasil implementasi dari *information retrieval* yang lebih kompleks.

Salah satu contoh mesin pencarian yang terbesar adalah Google, MSN Search, dan Yahoo!. Ketiganya adalah mesin pencarian berbasis *web* yang akan mencari berkas yang tersimpan dalam layanan www, ftp, publikasi milis, hingga *news group* dalam sebuah atau beberapa komputer yang terhubung dalam jaringan. Fungsi umum *search engine* dimaksudkan untuk mempermudah pengguna memperoleh informasi, juga untuk memenuhi beberapa kebutuhannya tanpa terbatas waktu dan tempat (Yazidanyastuti, 2011).

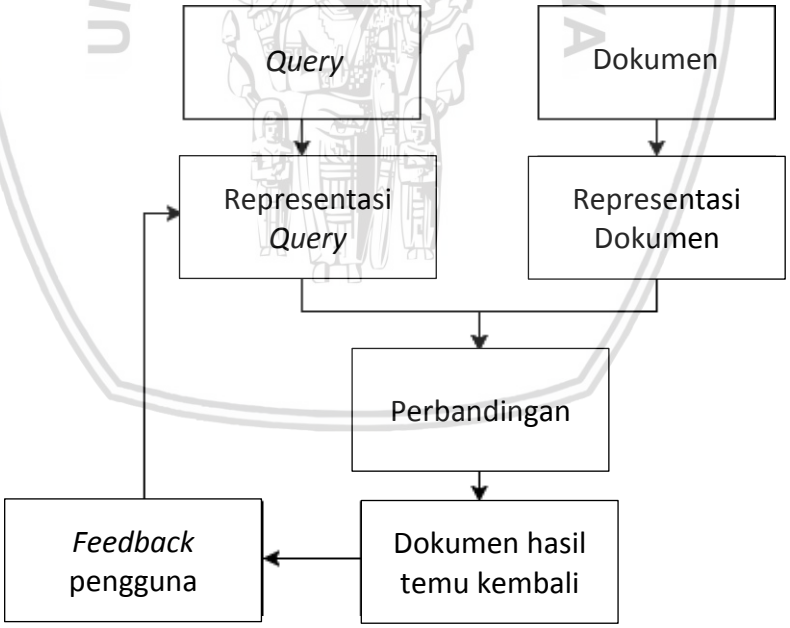
Mesin pencarian ini juga dimanfaatkan pada banyak aplikasi, hal ini dikarenakan mesin pencarian memudahkan pengguna untuk mencari informasi yang dibutuhkan dari situs lain. Terdapat tiga tugas dasar mesin pencari yang paling utama, yaitu:

a. mencari atau memilih bagian-bagian dari dokumen berdasar kata kunci
b. memberi indeks pada dokumen berdasar kata kunci yang diberikan pengguna
c. mengizinkan pengguna untuk mencari kata-kata atau kombinasi kata lain yang ditemukan pada indeks (Juliasari dan Sitompul, 2012, disitasi dalam Ahmadi, 2016).

2.3 Temu Kembali Informasi *(Information Retrieval)*

Temu kembali informasi adalah ilmu yang difokuskan untuk mempelajari prosedur dan metode untuk menemukan kembali informasi yang tersimpan dari berbagai sumber yang relevan dengan kata kunci yang dimasukkan pengguna.

Ada beberapa tahapan yang umum dalam proses menemukan kembali informasi ini, antara lain *indexing*, *searching*, dan *recalling*, secara umum digambarkan pada Gambar 2.1. Tujuannya untuk memenuhi kebutuhan informasi pengguna dengan menampilkan dokumen yang relevan dan mengurangi dokumen yang tidak relevan.



Gambar 2.1 Proses Umum Sistem *Information Retrieval*

Pemodelan *information retrieval* digunakan untuk menemukan representasi dokumen dan *query*, serta fungsi penilaiannya. Ada beberapa model IR yang biasanya digunakan, antara lain *exact match models*, *probabilistic models*, dan *vector space models.*

Exact match models adalah bentuk dasar dari kebanyakan sistem temu kembali. Model ini menentukan suatu dokumen akan diambil (ditemukan kembali) atau tidak sama sekali. Dokumen yang sudah ditemukan kembali tidak

akan diurutkan. Contoh pemodelannya pada model *boolean* dan model *region* (Hiemstra, 2009).

Model *probabilistic* memiliki suatu aturan yang menyatakan bahwa suatu dokumen harus ditemukan kembali jika nilai yang dihitung dari beberapa parameter yang sudah ditentukan kurang dari nilai batasnya (Losee, 1998). Contoh pemodelannya dapat digunakan pada tahapan *indexing,* model jaringan *bayesian*, dan model *page rank* milik Google (Hiemstra, 2009).

Vector space model memodelkan dokumen dan kata kunci sebagai vektor dalam suatu ruang vektor berdimensi tinggi dan menggunakan jarak antar dokumen dan kata kunci untuk menentukan persamaannya. Model ini akan digunakan dalam penelitian ini.

2.3.1 Vector Space Model

Vector space model (VSM) menggunakan vektor untuk merepresentasikan dokumen dan *query*. Vektornya dibentuk dari seluruh *term* atau kata-kata yang menyusun dokumen. Pada vektor dokumen dan *query*, tiap elemen merepresentasikan bobot kata-kata yang berhubungan dalam dokumen dan *query*-nya.

Bobot kata pada dokumen dan *query* dapat berupa biner (1 jika ada, 0 jika tidak ada kata yang dimaksud). Metode pembobotan biner ini mudah digunakan namun kurang akurat jika dibandingkan dengan metode pembobotan lain seperti pembobotan kata murni, pembobotan kata logaritmik, dan pembobotan kata normalisasi.

Pada dokumen dalam jumlah besar, skema metode yang paling banyak digunakan dengan hasil yang baik adalah metode pembobotan *TF-IDF*. *Tf* adalah *term frequency* atau frekuensi kemunculan suatu kata (*term*) dalam dokumen maupun *query*. Semakin sering suatu kata muncul dalam suatu dokumen maka semakin penting kata tersebut dalam dokumen. Pada persamaan *TF-IDF*, lazimnya digunakan nilai logaritma dari *tf* agar persebaran nilainya tidak terlalu jauh. Persamaannya dituliskan dalam Persamaan 2.1

$$w(tf) = \begin{cases} 1 + log_{10}\,(tf) & if\ tf > 0 \\ 0 & otherwise \end{cases} \quad (2.1)$$

Keterangan:

tf = frekuensi kemunculan kata dalam 1 dokumen

Idf adalah *inverse document frequency*, atau hasil kebalikan dari jumlah dokumen yang mengandung kata *t* dari keseluruhan dokumen. Jika suatu kata sering muncul dibanyak dokumen, semakin umum kata tersebut dan tidak bisa digunakan sebagai acuan untuk menggambarkan isi dokumen. Oleh karena itu, akan dicari kata-kata yang banyak muncul pada satu dokumen, namun jarang ditemukan pada dokumen lain sebagai representasi dokumen dan *query*. Rumus penghitungan *idf* dapat dituliskan dalam Persamaan 2.2.

$$idf\ (t_i) = \log \frac{N}{n(t_i)} \quad (2.2)$$

Keterangan:

t_i = kata

N = jumlah keseluruhan dokumen

$n(t_i)$= jumlah dokumen yang mengandung kata t_i

Nilai *TF-IDF* didapatkan dengan mengalikan nilai logaritma *tf* dan *idf*. Persamaannya dituliskan dalam Persamaan 2.3.

$$W(t,d) = w(tf) \times idf(t_i) \quad (2.3)$$

Setelah nilai *TF-IDF* dokumen dan *query* sudah didapatkan, nilai relevansi dokumen dan *query* dapat dihitung dengan mencocokan nilai *similarity* antara kedua vektor. Semakin besar nilai *similarity*-nya, semakin berkaitan dokumen dengan *query*. Salah satu metode untuk menghitung kemiripan yang paling sering digunakan adalah *cosine similarity.*

Cosine similarity adalah pengukuran kesamaan antara dua vektor bukan nol dari produk ruang dalam yang mengukur nilai kosinus sudut antara keduanya. Nilai kosinus dari 0° adalah 1 dan akan bernilai kurang dari 1 untuk sudut lain di antara jarak [0, 0,5π). Dengan demikian, nilainya bergantung pada arahnya, bukan besarnya. Dua vektor dengan arah yang sama memiliki nilai *cosine similarity* sebesar 1, dua vektor yang memiliki sudut 90° memiliki nilai *cosine similarity* sebesar 0, dan dua vektor yang saling bertolak belakang memiliki nilai *similarity* sebesar -1, terlepas dari besarnya vektor.

Nilai persamaan *cosine similarity* yang digunakan hanya pada ruang positif, hasilnya terletak di antara [0, 1]. Vektor-vektor yang mirip akan saling sejajar dan vektor-vektor yang berbeda akan saling tegak lurus. Dalam *information retrieval*, setiap *term* yang menyusun dokumen digambarkan dalam dimensi yang berbeda. Suatu dokumen akan direpresentasikan oleh suatu vektor yang nilainya dalam setiap dimensi sesuai dengan jumlah kemunculan *term* dalam dokumen.

Salah satu keuntungan dari pemanfaatan *cosine similarity* adalah kompleksitasnya rendah, khususnya untuk vektor-vektor yang tersebar, yang mana hanya perlu mempertimbangkan dimensi bukan nol. *Cosine similarity* dituliskan secara matematis dalam Persamaan 2.4.

$$sim\ (\overline{D}, \overline{Q}) = \frac{\overline{D} \cdot \overline{Q}}{|\overline{D}| \times |\overline{Q}|} \quad (2.4)$$

Keterangan:

$\overline{D}$ = Panjang vektor, didefinisikan sebagai $\overline{D} = \sum_{i=1}^{n} (d_i)^2$

$\overline{D} \cdot \overline{Q}$ = *Dot product* dari $\overline{D}$ dan $\overline{Q}$

2.4 Cross Language Information-Retrieval (CLIR)

Seiring dengan berjalannya waktu, kebutuhan pengguna akan informasi yang dulunya terhalang oleh batasan bahasa perlahan-lahan dapat dihilangkan. Pada tahun 2007, Google, mesin pencarian yang dibuat oleh Sergey Brin dan Larry Page 21 tahun yang lalu ini mengumumkan suatu fitur baru pada salah satu aplikasinya yakni *Google Translate*. Pada laman resmi Google, disebutkan bahwa salah satu tujuan di Google adalah untuk menyediakan akses informasi ke seluruh dunia dengan menghilangkan batasan bahasa yang menghalangi proses tersebut.

Google kemudian mengumumkan bahwa penggunanya sekarang dapat mencari dengan menggunakan kata kunci *(query)* dalam bahasanya dan mendapatkan hasil dalam bahasa yang lain. Misalnya, seseorang ingin mencari sebuah dokumen dalam bahasa Perancis, ia dapat memasukkan kata kunci dalam bahasa Indonesia dan hasilnya adalah dokumen yang relevan dalam bahasa Perancis dan Indonesia.

Dalam CLIR, ada 3 cara untuk melakukan penerjemahan:

1. melakukan penerjemahan representasi dokumen asli ke dalam bahasa pengguna;
2. melakukan penerjemahan *query* pengguna ke dalam bahasa yang lain;
3. melakukan penerjemahan dokumen dan *query*. Penerjemahan seperti ini disebut juga dengan *pivot language*.

Pada penelitian ini akan digunakan jenis penerjemahan *query*, yang mana pengguna bebas memilih bahasa (Inggris atau Indonesia) dan *query* yang dimasukkan akan diterjemahkan ke dalam bahasa yang lainnya. Pendekatan ini dipilih karena lebih fleksibel, efisien dan lebih mudah.

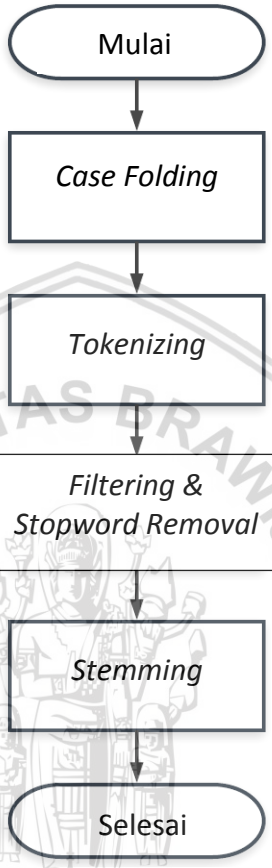
Pada penelitian milik Dan Wu dan Daqing He menjelaskan tentang penggunaan *Google Machine Translation System* untuk melakukan penerjemahan *query* dari bahasa Inggris ke bahasa Mandarin dan sebaliknya. Hasilnya menujukan bahwa penggunaan *Machine Translation* (MT) seperti Google lebih baik daripada penggunaan *Machine Readable Dictionary* (MRD) (Wu & He, 2010).

2.5 Text Mining

Text Mining adalah penerapan dari konsep dan teknik *data mining* untuk menemukan informasi yang bermanfaat untuk tujuan tertentu. *Text mining* juga dapat didefiniskan sebagai proses penambangan data berupa teks yang tidak terstruktur, seperti kutipan teks, dokumen PDF, dokumen Word, dan lain sebagainya. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan data teks tersebut. Sebelum memproses data teks, harus dilakukan satu tahapan untuk menormalisasi teks yang bentuknya tidak terstruktur. Tahapan tersebut disebut dengan *text preprocessing.*

2.5.1 Text Preprocessing

Text preprocessing adalah tahapan awal untuk mempersiapkan dokumen teks sebelum diolah lebih lanjut. Tahapan ini diperlukan untuk memudahkan pengolahan teks dilangkah selanjutnya. Secara umum, ada beberapa tahapan pada *text preprocessing* seperti pada Gambar 2.2.



Gambar 2.2 Diagram Alir *Text Preprocessing*

1. *Case Folding* adalah proses pengubahan abjad dalam dokumen menjadi huruf kecil (*lowercase*). Pada tahapan ini, semua karakter selain abjad juga akan dihapus dan dianggap sebagai *delimiter* atau pembatas. Contoh *case folding* dituliskan dalam Tabel 2.1.

Tabel 2.1 Contoh *Case Folding*

	Bahasa Inggris	Bahasa Indonesia
Masukan	*A computer is a device that can be instructed to carry out arbitrary sequences of arithmetic or logical operations automatically.*	Komputer adalah alat yang dipakai untuk mengolah data menurut prosedur yang telah dirumuskan.
Hasil	*a computer is a device that can be instructed to carry out arbitrary sequences of*	komputer adalah alat yang dipakai untuk mengolah data menurut prosedur yang telah

Tabel 2.1 Contoh *Case Folding* (lanjutan)

	Bahasa Inggris	Bahasa Indonesia
	arithmetic or logical operations automatically	dirumuskan

2. *Tokenizing* atau yang disebut juga tokenisasi adalah tahapan kedua dari *text preprocessing*. Pada tahapan ini, *string* masukan akan dipotong berdasar tiap kata penyusunnya. Untuk memecah *string* masukan, digunakan *whitespace* seperti spasi sebagai pembatas. Contoh hasil tokenisasi dituliskan pada Tabel 2.2.

Tabel 2.2 Contoh *Tokenizing*

	Bahasa Inggris	Bahasa Indonesia
Masukan	*a computer is a device that can be instructed to carry out arbitrary sequences of arithmetic or logical operations automatically*	komputer adalah alat yang dipakai untuk mengolah data menurut prosedur yang telah dirumuskan
Hasil	*a, computer, is, a, device, that, can, be, instructed, to, carry, out, arbitrary, sequences, of, arithmetic, or, logical, operations, automatically*	komputer, adalah, alat, yang, dipakai, untuk, mengolah, data, menurut, prosedur, yang, telah, dirumuskan

3. *Filtering* adalah tahap pengambilan kata-kata penting setelah *string* masukan dipisah. *Stopword removal* adalah proses penghapusan *stopword* atau kata-kata tidak penting yang ada pada dokumen. Dalam penelitian ini, daftar *stopword* yang digunakan adalah daftar *stopword* Tala[1] untuk bahasa Indonesia dan Stanford[2] untuk bahasa Inggris. Contoh *stopword* Bahasa Indonesia paling umum adalah *'yang', 'di', 'apa'*. *Stopword* pada bahasa Inggris misalnya *'the', 'where', 'a'*. Contoh hasil *filtering* dan *stopword removal* dituliskan pada Tabel 2.3.

Tabel 2.3 Contoh *Filtering*

	Bahasa Inggris	Bahasa Indonesia
Masukan	*a, computer, is, a, device,*	komputer, adalah, alat,

1.1.1

[1] F. Tala, "*A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*". Universiteit van Amsterdam, 2013.

[2] Stanfordnlp, 2018. *stanfordnlp/CoreNLP*. [online] GitHub. Tersedia pada: <https://github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt> [Diakses 17 April 2018].

Tabel 2.3 Contoh *Filtering* (lanjutan)

	Bahasa Inggris	Bahasa Indonesia
	that, can, be, instructed, to, carry, out, arbitrary, sequences, of, arithmetic, or, logical, operations, automatically	yang, dipakai, untuk, mengolah, data, menurut, prosedur, yang, telah, dirumuskan
Hasil	*computer, device, instructed, carry, arbitrary, sequences, arithmetic, logical, operations, automatically*	komputer, alat, dipakai, mengolah, data, prosedur, dirumuskan

4. *Stemming* adalah proses pengubahan kata-kata yang sudah difilter menjadi kata dasar. Semua imbuhan akan dihapus sehingga kata-kata yang menyusun dokumen akan menjadi kata dasar. Ada beberapa algoritme *stemming* untuk teks berbahasa Indonesia salah satunya adalah algoritme Nazief dan Adriani dan untuk Bahasa Inggris salah satunya adalah algoritme Porter. Contoh hasil dari proses *stemming* terdapat pada Tabel 2.4.

Tabel 2.4 Contoh *Stemming*

	Bahasa Inggris	Bahasa Indonesia
Masukan	*computer, device, instructed, carry, arbitrary, sequences, arithmetic, logical, operations, automatically*	komputer, alat, dipakai, mengolah, data, prosedur, dirumuskan
Hasil	*computer, device, instruct, carry, arbitrary, sequence, arithmetic, logical, operation, automatic*	komputer, alat, pakai, olah, data, prosedur, rumus

2.5.2 Algoritme *Stemming* Nazief dan Adriani

Dalam penelitian ini, algoritme *stemming* yang digunakan untuk kata berbahasa Indonesia adalah algoritme *stemming* Nazief dan Adriani. Berdasarkan penelitian yang dilakukan oleh Sahat dapat ditarik kesimpulan bahwa *stemming* dengan algoritme Nazief dan Adriani memiliki waktu yang singkat dengan nilai presisi lebih tinggi. (Sahat, 2017)

Algoritme *stemming* Nazief dan Adriani adalah sebagai berikut:

i. Cek kata di dalam kamus. Jika ditemukan, maka kata tersebut adalah kata dasar. Algoritme berhenti

ii. Hapus imbuhan infleks (*"-lah", "-kah", "-ku", "-mu", "-nya"*). Tahapan ini akan diulang jika menemukan partikel (*"-lah", "-kah", "-tah", "-pun"*). Dilakukan lagi pengecekan untuk menghapus kata ganti kepemilikan (*"-ku", "-mu", "-nya"*)

iii. Hapus sufiks turunan (*"-i", "-an", "-kan"*). Lakukan lagi pengecekan kata di kamus. Jika ditemukan, algoritme berhenti. Jika tidak, masuk ke langkah a:

   a. Jika *"-an"* telah dihapus dan huruf terakhir dari kata adalah huruf *"-k"*, maka *"k"* juga dihapus. Cek kata dalam kamus. Jika ditemukan, algoritme berhenti. Jika tidak, lanjut ke langkah b

   b. Akhiran yang sudah dihapus (*"-i", "-an", "-kan"*) dikembalikan lagi, dan lanjut ke langkah 4

iv. Hapus prefiks turunan. Jika pada langkah 3 ada sufiks turunan yang dihapus, maka algoritme lanjut ke langkah a. Jika tidak ada sufiks yang dihapus, lanjutkan ke langkah b:

   a. Cek tabel kombinasi awalan dan akhiran yang tidak diperbolehkan pada Tabel 2.5. Jika ditemukan, maka algoritme berhenti. Jika tidak, lanjutkan ke langkah b

   b. Untuk *i*=1 sampai 3, tentukan tipe awalan, kemudian hapus awalan. Jika kata dasar belum ditemukan dalam kamus, lakukan langkah 5. Jika sudah ditemukan, maka algoritme berhenti.

v. Bila dari langkah di atas masih belum ditemukan kata dasarnya di kamus, maka akan dilakukan analisis untuk pengecekan kombinasi awalan dan akhiran yang tidak diizinkan, seperti yang tertera dalam Tabel 2.5.

vi. Bila masih tidak ditemukan kata dasarnya, maka kata awal akan diasumsikan sebagai kata dasar. Proses *stemming* selesai.

Tabel 2.5 Tabel Awalan dan Akhiran yang Tidak Diizinkan pada Algoritme Nazief-Adriani

Awalan	Akhiran yang Tidak Diizinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan

2.5.3 Algoritme *Stemming* Porter

Stemming dokumen berbahasa Inggris akan menggunakan algoritme *stemming* Porter. Algoritme ini ditemukan oleh Martin Porter pada tahun 1980.

Mekanisme algoritme ini adalah mencari kata dasar suatu kata berimbuhan dengan membuang imbuhan pada kata-kata berbahasa Inggris. Pada tahun 1992, W.B. Frakes juga membuat beberapa penyesuaian untuk algoritme Porter agar sesuai dengan bahasa Indonesia.

Algoritme Porter menggunakan sistem perubahan sufiks S1 menjadi S2 jika memenuhi kondisi tertentu, seperti yang tercantum dalam Tabel 2.6. Persamaannya ditujukan pada Persamaan 2.5.

$$(condition)\ S1 \rightarrow S2 \tag{2.5}$$

Tabel 2.6 Aturan Perubahan untuk *Stemming* Porter

Langkah	Kondisi	S1 → S2
1a		*SSES → SS* *IES → I* *SS → SS* *S →*
1b	(m > 0) (*v*) (*v*)	*EED → EE* *ED →* *ING →*
1c	(*v*)	*Y → I*
2	(m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0) (m>0)	*ATIONAL→ ATE* *TIONAL → TION* *ENCI → ENCE* *ANCI → ANCE* *IZER → IZE* *ABLI → ABLE* *ALLI → AL* *ENTLI → ENT* *ELI → E* *OUSLI → OUS* *IZATION →IZE* *ATION → ATE* *ATOR → ATE* *ALISM →AL* *IVENESS → IVE* *FULNESS → FUL*

Tabel 2.6 Aturan Perubahan untuk *Stemming* Porter (lanjutan)

Langkah	Kondisi	S1 → S2
	(m>0)	*OUSNESS → OUS*
	(m>0)	*ALITI → AL*
	(m>0)	*IVITI → IVE*
	(m>0)	*BILITI → BLE*
3	(m>0)	*ICATE → IC*
	(m>0)	*ATIVE →*
	(m>0)	*ALIZE → AL*
	(m>0)	*ICITI → IC*
	(m>0)	*ICAL → IC*
	(m>0)	*FUL →*
	(m>0)	*NESS →*
4	(m>1)	*AL →*
	(m>1)	*ANCE →*
	(m>1)	*ENCE →*
	(m>1)	*ER →*
	(m>1)	*IC →*
	(m>1)	*ABLE →*
	(m>1)	*IBLE →*
	(m>1)	*ANT →*
	(m>1)	*EMENT →*
	(m>1)	*MENT →*
	(m>1)	*ENT →*
	(m>1 DAN (*S ATAU *T))	*ION →*
	(m>1)	*OU →*
	(m>1)	*ISM →*
	(m>1)	*ATE →*
	(m>1)	*ITI →*
	(m>1)	*OUS →*
	(m>1)	*IVE →*
	(m>1)	*IZE →*
5a	(m>1)	*E →*

Tabel 2.6 Aturan Perubahan untuk *Stemming* Porter (lanjutan)

Langkah	Kondisi	S1 → S2
	(m=1 DAN TIDAK *o)	*E* →
5b	(m > 1 DAN *d DAN *L)	→ *single letter*

2.6 Pengujian dan Analisis

Pengujian dan analisis adalah tahap terakhir dari pembuatan sistem. Pengujian dilakukan untuk mengetahui jika sistem yang dibuat sudah sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya. Pengujian dilakukan dengan menghtung *Precision@ K (P@K)* untuk mengukur ketepatan sistem dalam menemukan dokumen relevan yang sesuai dengan masukan pengguna. Pengujian pengaruh penggunaan *query* yang sudah diterjemahkan dengan *query* yang tidak diterjemahkan dilakukan dengan membandingkan hasil *Precision-Recall Curve*-nya.

2.6.1 Precision @ K (P@K)

Precision adalah perbandingan jumlah dokumen relevan yang ditemukan kembali dengan seluruh dokumen yang dianggap relevan. *Precision @ K* menghitung nilai presisi dalam sejumlah *top-k* dokumen. Misalnya, 8 dari *top-10* hasil pencarian relevan dengan kebutuhan informasi pengguna, maka nilai presisinya sebesar 0,8. Rumus *precision @ K* dituliskan secara matematis dalam Persamaan 2.6.

$$\boldsymbol{Precision@k = \frac{jumlah\ dokumen\ relevan\ dalam\ top\ k}{jumlah\ seluruh\ dokumen\ dalam\ k}} \quad \textbf{(2.6)}$$

2.6.2 Precision Recall Curve (PRC)

Precision-Recall Curve (PRC) adalah sebuah kurva yang digunakan untuk menggambarkan hubungan antara *precision* dan *recall,* serta digunakan untuk menghitung nilai *precision* pada beberapa tingkat *recall.*

Precision adalah perbandingan dokumen relevan yang ditemukan kembali dengan keseluruhan dokumen yang ditemukan kembali. *Precision* juga dapat diartikan sebagai kecocokan antara permintaan informasi dan jawaban atas permintaan tersebut. Secara matematis, *precision* dapat dituliskan seperti pada Persamaan 2.7.

$$Precision = \frac{|dokumen\ relevan \cap dokumen\ ditemukan\ kembali|}{|dokumen\ ditemukan\ kembali|} \quad \textbf{(2.7)}$$

Recall adalah perbandingan dokumen relevan yang ditemukan kembali dengan seluruh dokumen relevan. Pada *PRC*, jumlah dokumen relevan dianggap sebanyak jumlah maksimal dokumen yang ditemukan kembali. Rumusnya secara matematis dituliskan pada Persamaan 2.8

$$Recall = \frac{|dokumen\ relevan \cap dokumen\ ditemukan\ kembali|}{|dokumen\ relevan|} \quad \textbf{(2.8)}$$

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan masing-masing tahapan dalam penelitian untuk menyelesaikan penelitian tentang pembuatan Sistem Pencarian Jurnal Ilmiah *Cross Language*.

3.1 Tipe Penelitian

Penelitian yang dilakukan bersifat non implementatif. Penelitian non implementatif berfokus pada pengamatan terhadap suatu fenomena atau situasi tertentu yang akan menghasilkan tinjauan ilmiah. Penelitian non implemetatif juga dapat berupa pelaksanaan tinjauan dari sebuah ilmu yang sudah ada.

Pendekatan yang digunakan adalah pendekatan analitis. Pendekatan ini akan menjelaskan hubungan antar elemen dalam objek penelitian dengan fenomena yang sedang diteliti. Hasil kegiatannya akan menjawab pertanyaan-pertanyaan dalam rumusan masalah yang sudah ditentukan di awal.

3.2 Kajian Pustaka

Tahapan pertama yang akan dilakukan adalah pengkajian pustaka yang berhubungan dengan penelitian. Pustaka yang digunakan di antaranya adalah rujuan penelitian sebelumnya, jurnal ilmiah, dan buku. Bidang-bidang yang dikaji antara lain:

1. Mesin pencarian *(search engine)*
2. *Text mining*
3. Temu kembali informasi *(information retrieval)*
4. *Cross language information retrieval*
5. *Vector space model (VSM)*

3.3 Pengumpulan Data

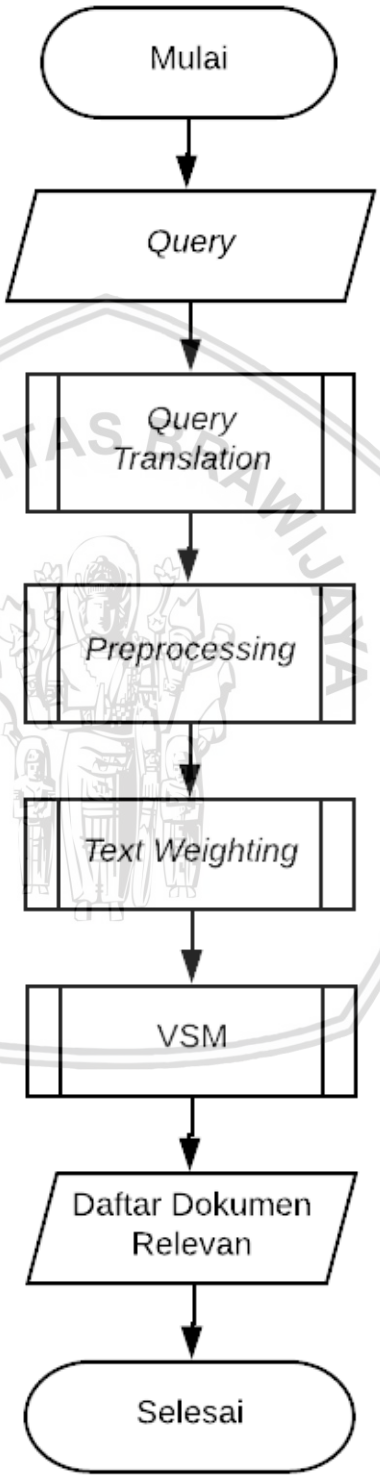
Pengumpulan data digunakan untuk mengumpulkan jurnal-jurnal dari perpustakaan jurnal online seperti Science Direct[3] dan Direktori JTIIK FILKOM UB[4]. Teknik yang digunakan untuk pengumpulan data ini adalah metode penelitian kepustakaan, yaitu suatu metode yang menggunakan internet untuk mengakses *website* penyedia jurnal ilmiah. Dokumen yang akan diambil adalah dokumen jurnal berbahasa Inggris dan Indonesia dari bidang ilmu komputer.

1.1.1

[3] www.sciencedirect.com
[4] jtiik.ub.ac.id

3.4 Perancangan Algoritme

Untuk mengimplementasikan proses *cross language information retrieval* ada beberapa tahapan yang harus dilakukan. Tahapan-tahapan tersebut seperti yang tertera pada Gambar 3.1



Gambar 3.1 Diagram Alir Proses *Cross Language Information Retrieval*

Dari *flowchart* pada Gambar 3.1, proses dapat dijabarkan sebagai berikut:

1. Pengguna memasukkan *query*. *Query* akan diterjemahkan sebelum dilakukan proses *preprocessing*
2. *Query* masukkan dan *query* terjemahan akan di-*preprocessing* sesuai dengan bahasanya
3. Setelah didapatkan *term*-nya, bobot *query* akan dihitung
4. Sistem akan mengukur bobot *query* dengan indeks dokumen dengan menggunakan metode VSM
5. Hasilnya adalah daftar dokumen yang relevan menurut sistem, diurutkan dari dokumen yang paling mirip dengan *query*.

3.5 Implementasi Sistem

Untuk mengimplementasikan sistem yang telah dirancang sebelumnya, digunakan lingkungan sistem dengan spesifikasi sebagai berikut:

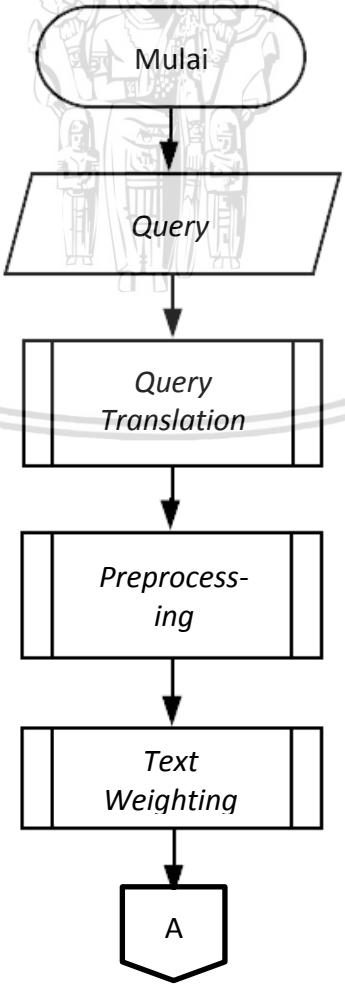
1. Perangkat Lunak
   a. Sistem operasi Microsoft Windows 7 Ultimate
   b. Python versi 2.7.13
   c. IDE Spyder versi 3.1.3
2. Perangkat Keras
   a. *Processor* Intel Celeron 1007U
   b. RAM 8GB
   c. *Harddisk* 320GB

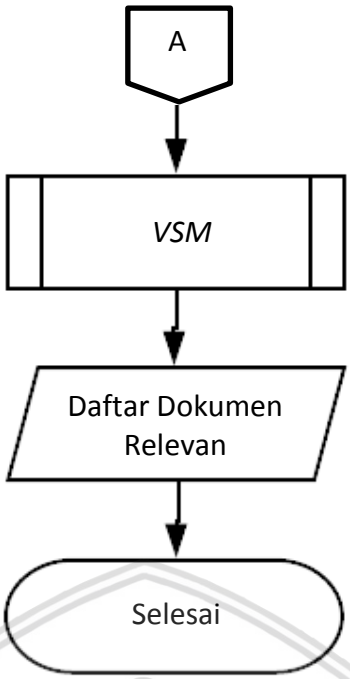
BAB 4 PERANCANGAN DAN IMPLEMENTASI

4.1 Deskripsi Umum

Sistem pencarian jurnal ilmiah dengan menggunakan *VSM* adalah sistem yang dapat menemukan kembali jurnal-jurnal ilmiah dalam bahasa Indonesia dan bahasa Inggris sesuai dengan *query* atau kata kunci yang dimasukkan pengguna. Data-data yang digunakan dalam penelitian ini diambil dari berbagai sumber, antara lain dari Direktori JTIIK FILKOM UB, Science Direct, dan IEEE. Dari dokumen-dokumen tersebut akan diambil *term* yang dapat menggambarkan isi dokumen. Bagian yang akan digunakan adalah bagian abstrak dan *keywords* dari dokumen jurnal ilmiah.

Sebelum diproses, *term* dari dokumen dan *query* akan di-*preprocessing* terlebih dahulu. Setelah didapatkan bagian yang penting, *query* akan diterjemahkan ke bahasa target, jika *query* dalam Bahasa Inggris, maka bahasa target adalah Bahasa Indonesia, berlaku juga sebaliknya. *Term* dokumen akan dihitung bobotnya, kemudian dicocokan dengan *query* dari pengguna dengan menggunakan metode *VSM*. Hasilnya adalah dokumen-dokumen yang dianggap relevan oleh sistem dan diurutkan berdasarkan relevansinya terhadap *query*. Tahap perancangan sistem secara umum ditunjukan pada Gambar 4.1.

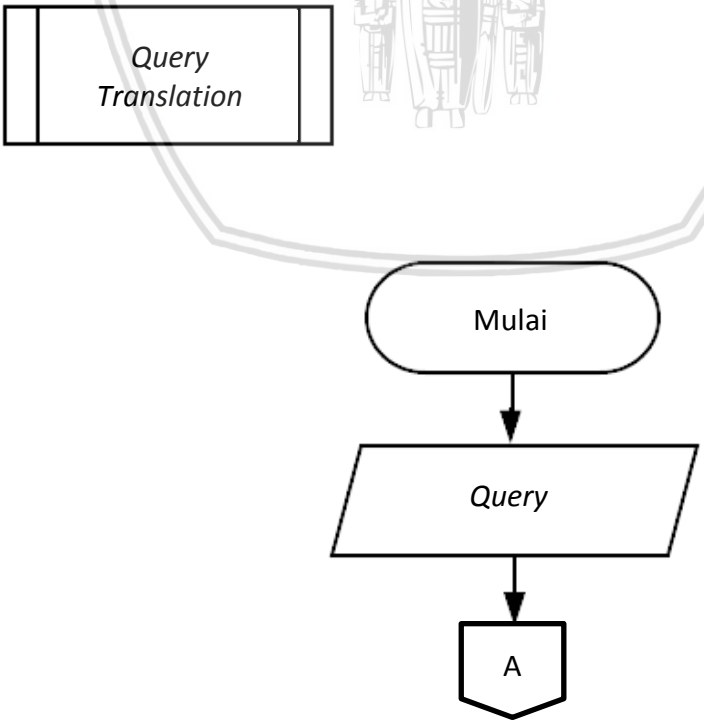


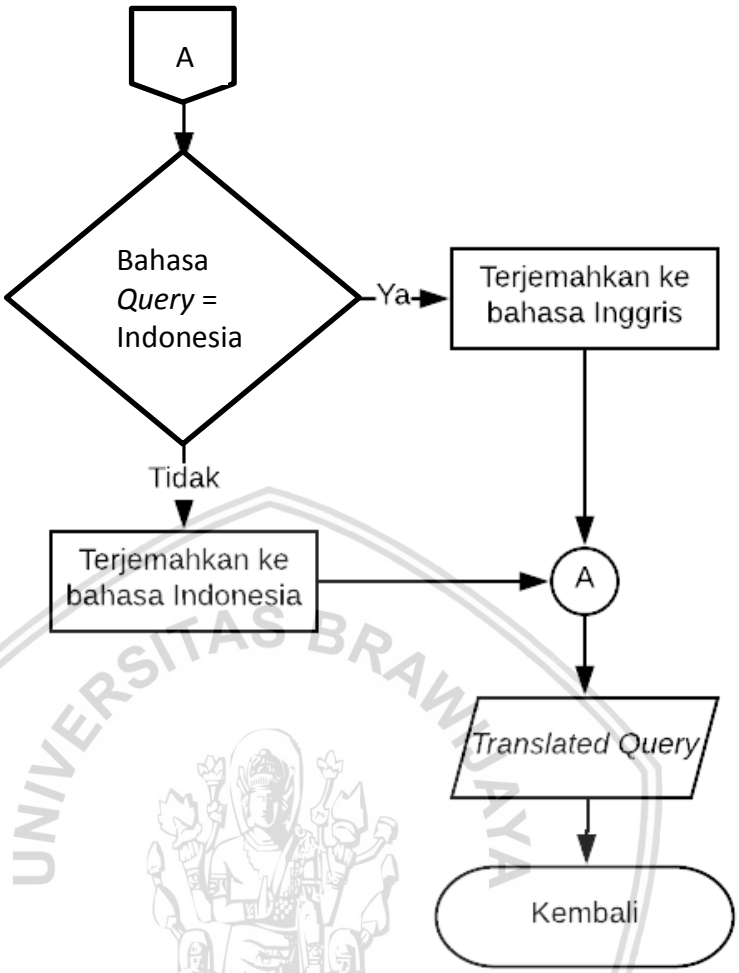


Gambar 4.1 Diagram Alir Perancangan Umum Sistem

4.2 *Query Translation*

Untuk memulai pencarian jurnal, pengguna akan diminta memasukkan *query* dalam bahasa Indonesia atau bahasa Inggris. *Query* ini akan diterjemahkan ke dalam bahasa yang lain sehingga didapatkan *query* dalam 2 bahasa, yakni bahasa Indonesia dan bahasa Inggris. Tahapan penerjemahan *query* ini dijelaskan dalam bentuk diagram alir pada Gambar 4.2.

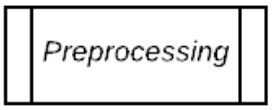


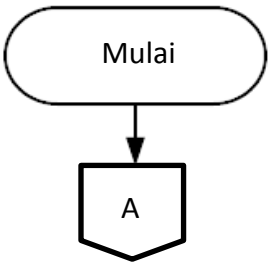


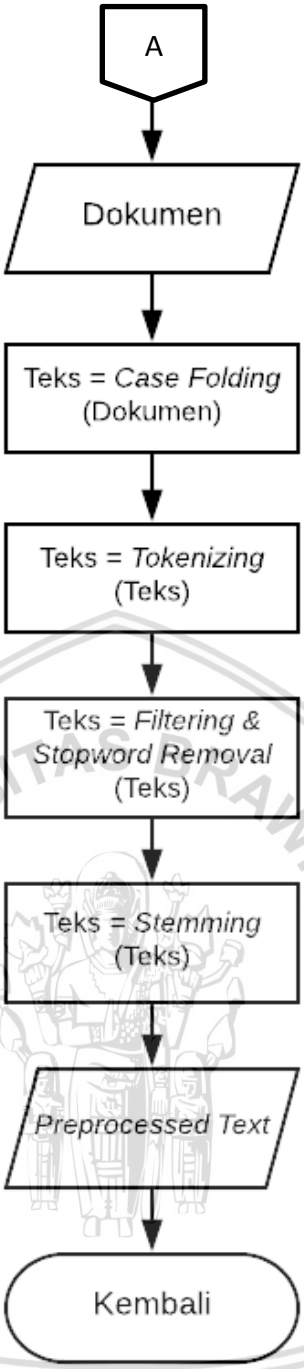
Gambar 4.2 Diagram Alir *Query Translation*

4.3 Text Preprocessing

Text preprocessing adalah tahapan awal untuk membentuk *term* dari dokumen teks yang tidak terstruktur. *Text preprocessing* dilakukan pada *query* masukan pengguna dan kumpulan dokumen. Hasil dari *text preprocessing* adalah teks yang sudah bersih dari simbol dan karakter selain alfabet serta *stopword* dan imbuhan. Ada 4 tahapan utama yang dilakukan dalam *preprocessing* dan dapat diuraikan seperti pada Gambar 4.3.







Gambar 4.3 Diagram Alir *Text Preprocessing*

4.3.1 *Case Folding*

Case folding adalah bagian dari *preprocessing*. Dalam tahapan ini, seluruh alfabet dalam dokumen akan diubah menjadi huruf kecil (*lowercase*). Seluruh karakter selain alfabet akan dihapus dan dianggap sebagai pembatas. Keluarannya adalah kalimat yang tersusun dari alfabet-alfabet dalam huruf kecil.

4.3.2 Tokenizing

Tokenizing atau tokenisasi adalah tahapan kedua dari *text preprocessing*. Pada tahapan ini, *string* masukan akan dipotong berdasar tiap kata penyusunnya. Untuk memecah *string* masukan, digunakan *whitespace* seperti spasi sebagai pembatas. *Output*-nya adalah kata-kata atau *token* yang menyusun dokumen.

4.3.3 Filtering & Stopword Removal

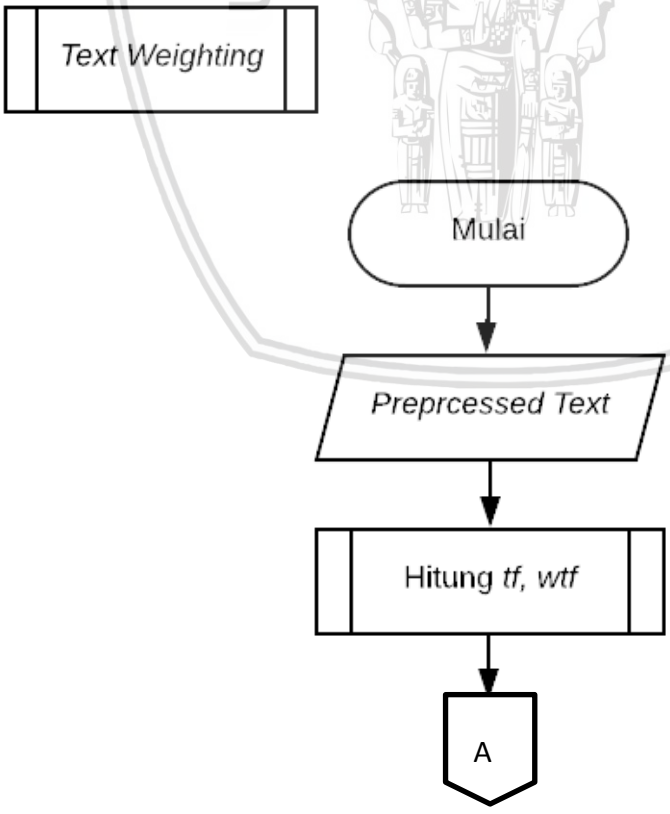
Filtering adalah proses pengambilan kata-kata atau *token* yang dianggap penting setelah *string* masukan dipisah. *Stopword removal* adalah proses penghapusan kata-kata tidak penting yang ada pada dokumen. Keluaran dari tahapan ini adalah token-token yang dianggap bisa menggambarkan isi dokumen.

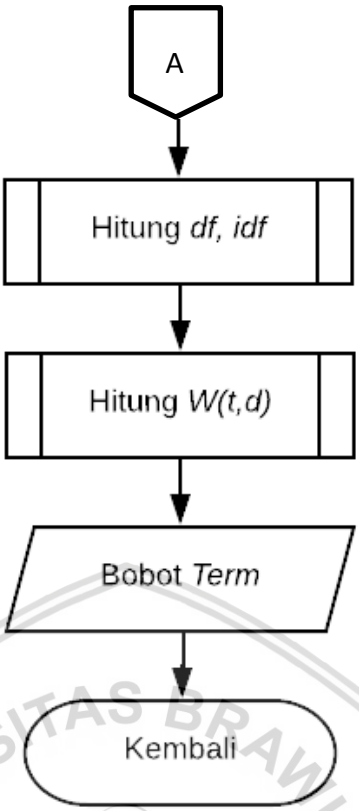
4.3.4 *Stemming*

Stemming merupakan proses terakhir dalam *preprocessing*, yakni pengubahan kata-kata yang sudah difilter menjadi kata dasar. Semua imbuhan akan dihapus sehingga kata-kata yang menyusun dokumen akan menjadi kata dasar. Hal ini dilakukan untuk mempermudah pembobotan dan meningkatkan hasilnya. *Output* dari *stemming* adalah *term* yang siap diproses untuk dihitung bobotnya.

4.4 Text Weighting

Pembobotan teks adalah tahapan untuk memberi nilai tiap-tiap *term* yang menyusun dokumen dan *query* agar dapat diproses lebih lanjut. Dalam tahap ini, setidaknya ada 3 proses yang dilakukan, yaitu penghitungan nilai *tf* dan *df*, penghitungan nilai *w(tf)* dan *idf*, penghitungan nilai *W(t,d)*. Secara umum, tahap pembobotan teks dapat digambarkan seperti pada Gambar 4.4.

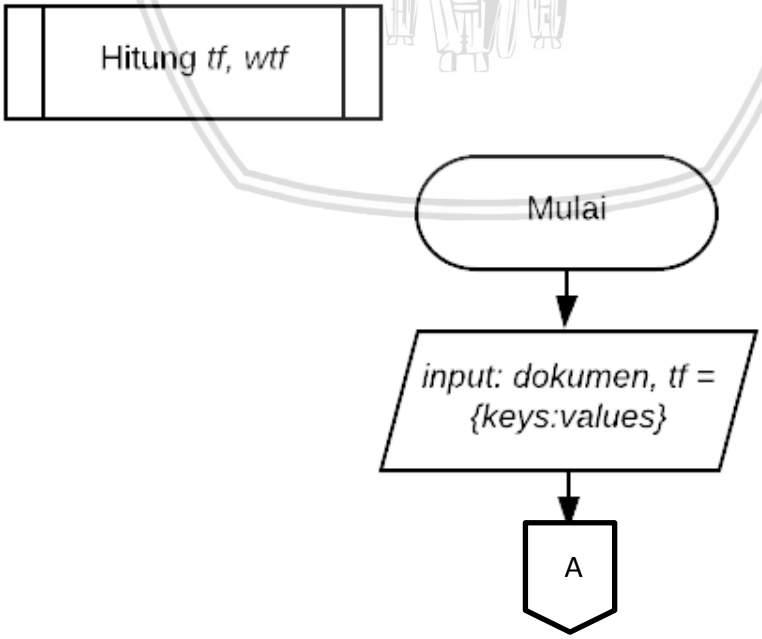




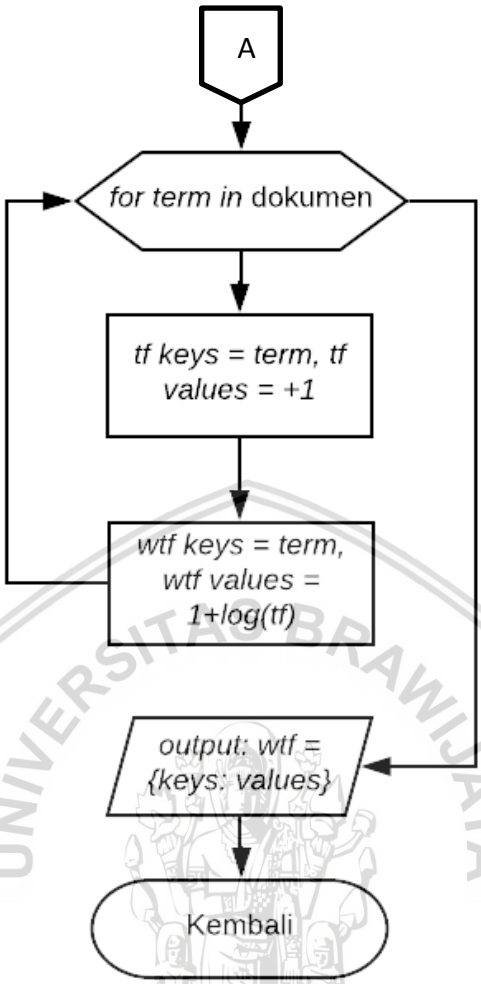
Gambar 4.4 Diagram Alir *Text Weighting*

4.4.1 Penghitungan Nilai *tf* dan *wtf*

Tf adalah frekuensi kemunculan *term t* dalam dokumen *d*. *Wtf* adalah hasil logaritma dari nilai *tf*. Nilai *wtf* akan digunakan untuk pembobotan *term* ditahap selanjutnya. Diagram alir penghitungan nilai *tf* dan *wtf* digambarkan pada Gambar 4.5.



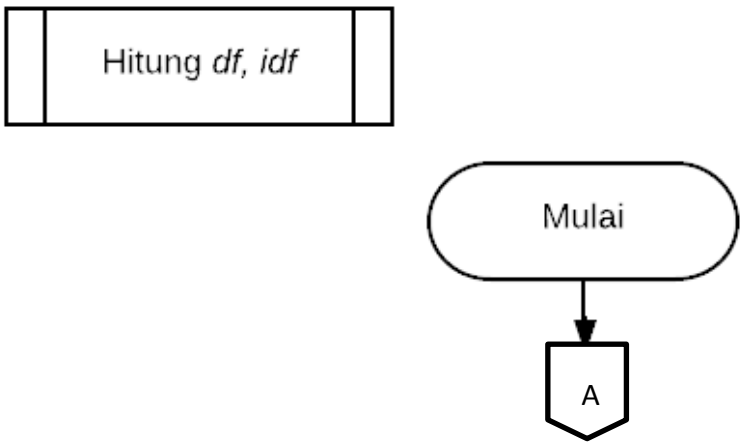
.

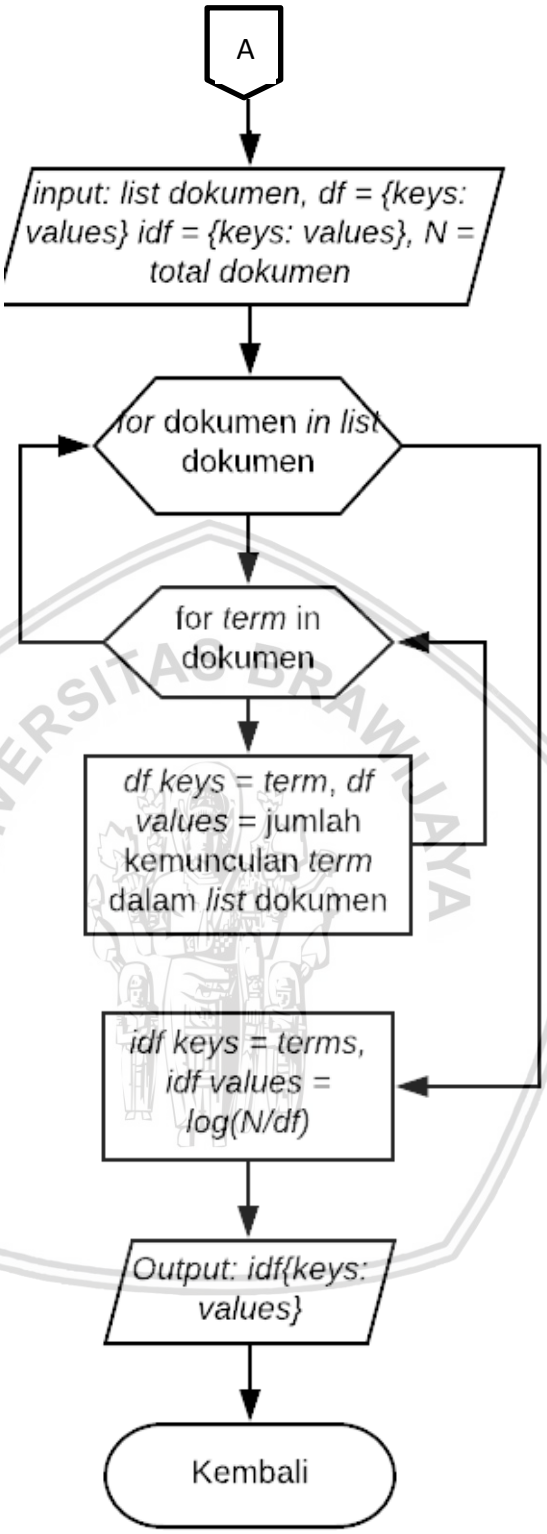


Gambar 4.5 Diagram Alir Penghitungan Nilai *tf* dan *wtf*

4.4.2 Penghitungan Nilai *df* dan *idf*

Nilai *df* adalah jumlah kemunculan term dalam keseluruhan dokumen. *Idf* adalah nilai invers *df*. Nilai *idf* akan digunakan untuk pembobotan term ditahap selanjutnya. Diagram alir penghitungannya digambarkan pada **Gambar 4.6**.

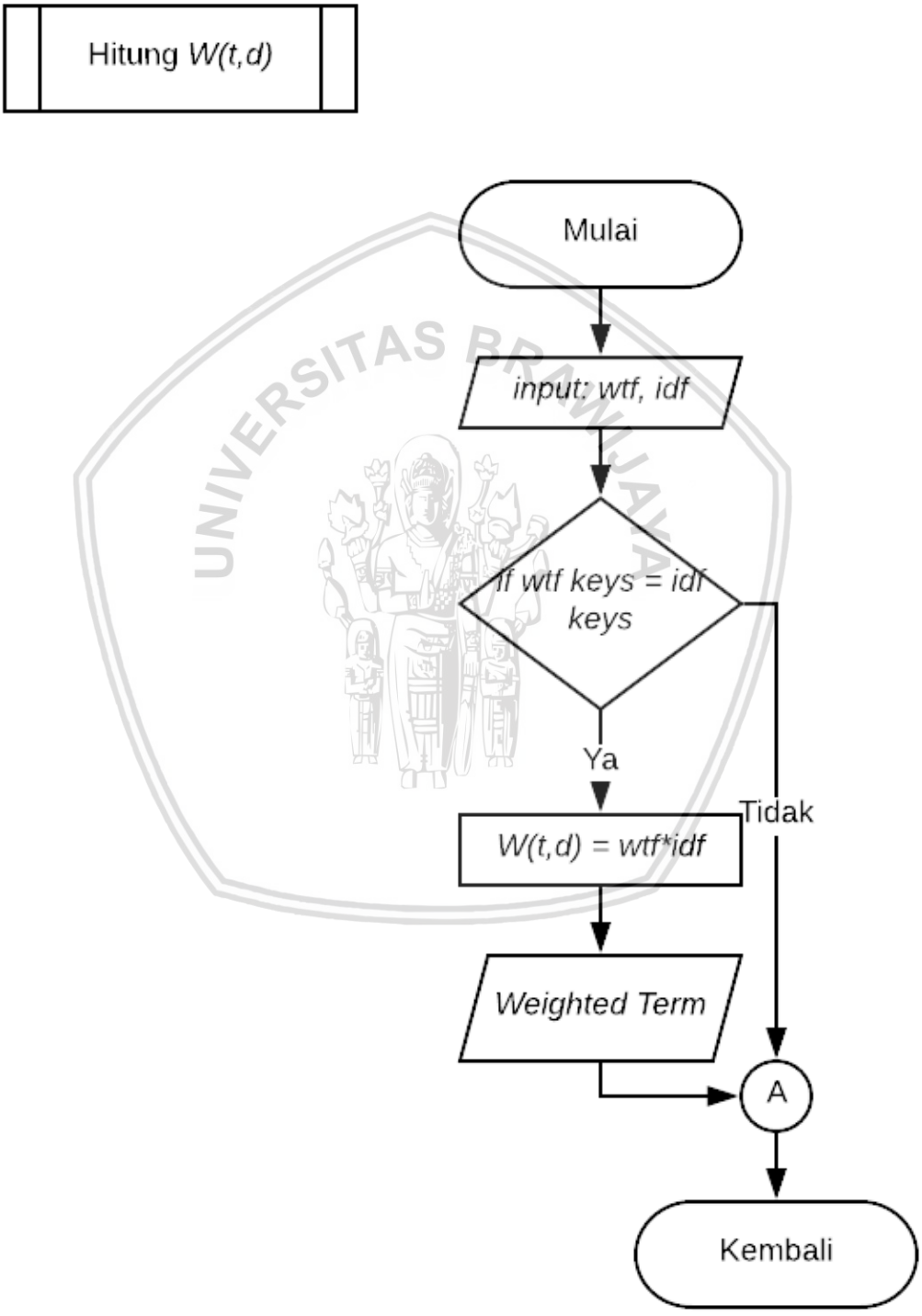




Gambar 4.6 Diagram Alir Penghitungan Nilai *df* dan *idf*

4.4.3 Penghitungan nilai *W(t,d)*

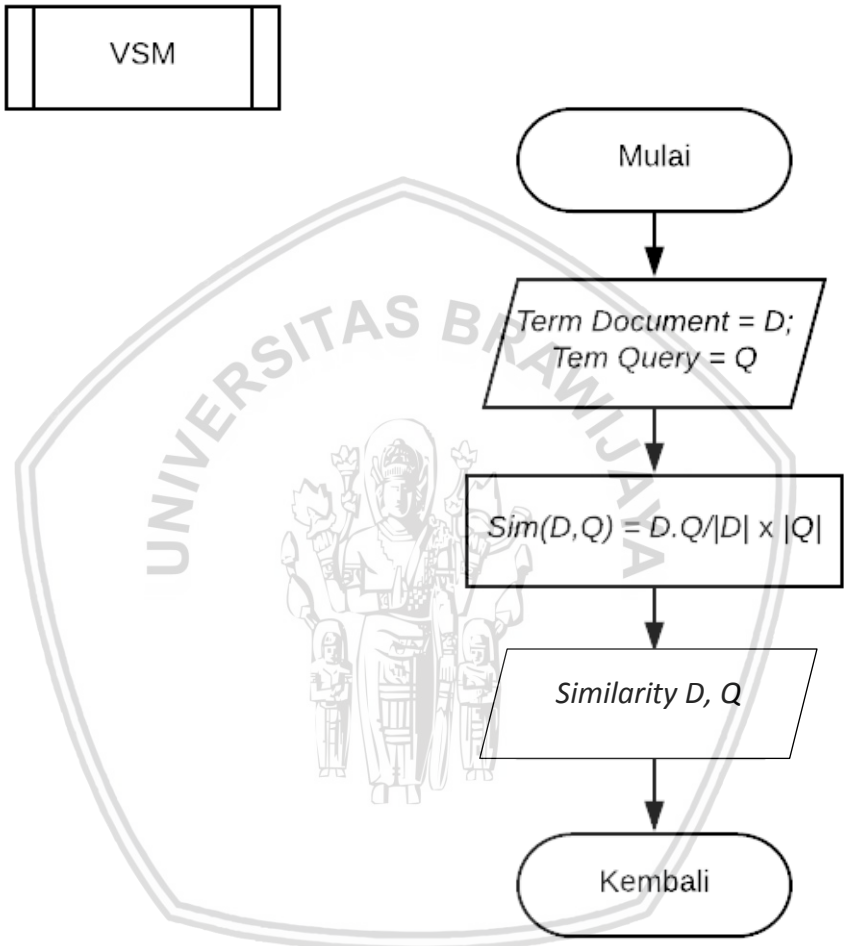
W(t,d) disebut juga dengan *TF.IDF* adalah hasil perkalian dari *w(tf)* dan *idf* dari tahap sebelumnya. Hasil perkalian tersebut merupakan bobot *term* yang dapat digunakan untuk mengukur kemiripan dokumen dengan *query*. Diagram alirnya tertera pada Gambar 4.7.



Gambar 4.7 Diagram Alir Penghitungan Nilai *W(t,d)*

4.5 Vector Space Model

Vector space model adalah metode yang digunakan untuk mengukur kemiripan dua dokumen dengan mengukur jarak kosinus antar vektor dalam suatu ruang vektor berdimensi banyak. Bobot yang didapatkan dari tahapan sebelumnya akan dihitung jarak kosinusnya dan diurutkan dari dokumen yang memiliki nilai *cosine similarity* tertinggi dengan *query*. Diagram alir tahapan VSM digambarkan pada Gambar 4.8.



Gambar 4.8 Diagram Alir Metode VSM

4.6 Manualisasi Metode *Vector Space Model*

Pada penghitungan manual metode VSM digunakan contoh berupa 10 dokumen kutipan dari jurnal ilmiah dengan pembagian 5 dokumen berbahasa Indonesia dan 5 dokumen berbahasa Inggris. Dokumen-dokumen tersebut dituliskan dalam Tabel 4.1. *Query* yang digunakan adalah "*seleksi fitur*" yang di-*input* dalam bahasa Indonesia. *Query* dianggap sudah melewati proses terjemahan, sehingga sudah ada 2 *query* dalam bahasa Indonesia dan bahasa Inggris. Dua *query* ini dianggap sebagai dokumen baru dan dimasukkan juga ke dalam Tabel 4.1.

Tabel 4.1 Tabel Dokumen Kutipan Jurnal Ilmiah

No. Dokumen	Isi dokumen
D1	Klasifikasi dapat diterapkan disemua bidang kehidupan termasuk dalam teks.
D2	Algoritma klasifikasi menggunakan semua fitur yang terdapat pada data untuk membangun sebuah model.
D3	Padahal, tidak semua fitur tersebut sesuai terhadap hasil klasifikasi.
D4	Seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan.
D5	Seleksi fitur bertujuan untuk memilih fitur terbaik dari suatu kumpulan data fitur
D6	*Feature Selection aims to select the best features of a dataset of features.*
D7	*The purpose of this research is to apply the information gain method in the feature selection system for Indonesian language text documents.*
D8	*Information Gain method is a method that using scoring techniques for weighting a feature by using a maximum entrophy.*
D9	*Selected features is featured with information gain values greater than or equal to a certain threshold value.*
D10	*The threshold value used is 0,02; 0,05; and 0,07.*
Q-ID	Seleksi fitur.
Q-EN	Feature selection

4.6.1 Preprocessing

Tahapan pertama yang harus dilakukan dalam manualisasi ini adalah *preprocessing*. Hasil dari setiap subproses yang dilakukan dalam *preprocessing* ditunjukan pada Tabel 4.2 sampai Tabel 4.5.

1. *Case folding*

   Case folding akan mengubah seluruh huruf kapital dalam kalimat menjadi huruf kecil dan menghapus tanda baca serta karakter lain selain alfabet menjadi pembatas.

Tabel 4.2 Tabel Hasil *Case Folding*

No. Dokumen	Isi dokumen
D1	Klasifikasi dapat diterapkan disemua bidang kehidupan termasuk dalam teks.

Tabel 4.2 Tabel Hasil *Case Folding* (lanjutan)

No. Dokumen	Isi dokumen
	klasifikasi dapat diterapkan disemua bidang kehidupan termasuk dalam teks
D2	Algoritma klasifikasi menggunakan semua fitur yang terdapat pada data untuk membangun sebuah model.
	algoritma klasifikasi menggunakan semua fitur yang terdapat pada data untuk membangun sebuah model
D3	Padahal, tidak semua fitur tersebut sesuai terhadap hasil klasifikasi.
	padahal tidak semua fitur tersebut sesuai terhadap hasil klasifikasi
D4	Seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan.
	seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan
D5	Seleksi fitur bertujuan untuk memilih fitur terbaik dari suatu kumpulan data fitur
	seleksi fitur bertujuan untuk memilih fitur terbaik dari suatu kumpulan data fitur
D6	*Feature Selection aims to select the best features of a dataset of features.*
	feature selection aims to select the best features of a dataset of features
D7	*The purpose of this research is to apply the information gain method in the feature selection system for Indonesian language text documents.*
	the purpose of this research is to apply the information gain method in the feature selection system for indonesian language text documents
D8	*Information Gain method is a method that using scoring techniques for weighting a feature by using a maximum entrophy.*
	information gain method is a method that using scoring techniques for weighting a feature by using a maximum entrophy
D9	*Selected features is featured with information gain values greater than or equal to a certain threshold value.*
	selected features is featured with information gain values greater than or equal to a certain threshold value
D10	*The threshold value used is 0,02; 0,05; and 0,07*

Tabel 4.2 Tabel Hasil *Case Folding*(lanjutan)

No. Dokumen	Isi dokumen
	the threshold value used is and
Q-ID	Seleksi fitur
	seleksi fitur
Q-EN	Feature selection
	feature selection

2. *Tokenizing*

   Hasil dari *case folding* akan dipecah menjadi token yang menggambarkan isi dokumen.

Tabel 4.3 Tabel Hasil *Tokenizing*

No. Dokumen	Isi dokumen
D1	klasifikasi dapat diterapkan disemua bidang kehidupan termasuk dalam teks
	klasifikasi, dapat, diterapkan, disemua, bidang, kehidupan, termasuk, dalam, teks
D2	algoritma klasifikasi menggunakan semua fitur yang terdapat pada data untuk membangun sebuah model
	algoritma, klasifikasi, menggunakan, semua, fitur, yang, terdapat, pada, data, untuk, membangun, sebuah, model
D3	padahal tidak semua fitur tersebut sesuai terhadap hasil klasifikasi
	padahal, tidak, semua, fitur, tersebut, sesuai, terhadap, hasil, klasifikasi
D4	seleksi fitur adalah teknik untuk memilih fitur penting dan relevan terhadap data dan mengurangi fitur yang tidak relevan
	seleksi, fitur, adalah, teknik, untuk, memilih, fitur, penting, dan relevan, terhadap, data, dan, mengurangi, fitur, yang, tidak, relevan.
D5	seleksi fitur bertujuan untuk memilih fitur terbaik dari suatu kumpulan data fitur
	seleksi, fitur, bertujuan, untuk, memillih, fitur, terbaik, dari, suatu, kumpulan, data, fitur
D6	*feature selection aims to select the best features of a dataset of features*
	feature, selection, aims, to, select, the, best, features, of, a ,

Tabel 4.3 Tabel Hasil *Tokenizing* (lanjutan)

No. Dokumen	Isi dokumen
	dataset, of, features
D7	*the purpose of this research is to apply the information gain method in the feature selection system for indonesian language text documents*
	the, purpose, of, this, research, is, to, apply, the, information, gain, method, in, the, feature, selection, system, for, indonesian, language, text, documents
D8	*information gain method is a method that using scoring techniques for weighting a feature by using a maximum entrophy*
	information, gain, method, is, a, method, that, using, scoring, techniques, for, weighting, a, feature, by, using, a, maximum, entrophy
D9	*selected features is featured with information gain values greater than or equal to a certain threshold value*
	selected, features, is, featured, with, information, gain, values, greater, than, or, equal, to, a, certain, threshold, value
D10	*the threshold value used is and*
	the, threshold, value, used, is, and
Q-ID	seleksi fitur
	seleksi, fitur
Q-EN	feature selection
	feature, selection

3. *Filtering & Stopword Removal*

   Token yang didapat dari tahap *tokenizing* akan difilter dan dihapus *stopword*-nya.

Tabel 4.4 Tabel Hasil *Filtering & Stopword Removal*

No. Dokumen	Isi dokumen
D1	klasifikasi, dapat, diterapkan, disemua, bidang, kehidupan, termasuk, dalam, teks
	klasifikasi, diterapkan, bidang, kehidupan, termasuk, teks
D2	algoritma, klasifikasi, menggunakan, semua, fitur, yang, terdapat, pada, data, untuk, membangun, sebuah, model
	algoritma, klasifikasi, menggunakan, fitur, data, membangun, model
D3	padahal, tidak, semua, fitur, tersebut, sesuai, terhadap, hasil, klasifikasi

Tabel 4.4 Tabel Hasil *Filtering & Stopword Removal* (lanjutan)

No. Dokumen	Isi dokumen
	fitur, sesuai, hasil, klasifikasi
D4	seleksi, fitur, adalah, teknik, untuk, memilih, fitur, penting, dan relevan, terhadap, data, dan, mengurangi, fitur, yang, tidak, relevan.
	seleksi, fitur, teknik, memilih, fitur, penting, relevan, data, mengurangi, fitur, relevan.
D5	seleksi, fitur, bertujuan, untuk, memillih, fitur, terbaik, dari, suatu, kumpulan, data, fitur
	seleksi, fitur, bertujuan, memillih, fitur, terbaik, kumpulan, data, fitur
D6	*feature, selection, aims, to, select, the, best, features, of, a , dataset, of, features*
	feature, selection, aims, select, best, features, dataset, features
D7	*the, purpose, of, this, research, is, to, apply, the, information, gain, method, in, the, feature, selection, system, for, indonesian, language, text, documents*
	purpose, research, apply, information, gain, method, feature, selection, system, indonesian, language, text, documents
D8	*information, gain, method, is, a, method, that, using, scoring, techniques, for, weighting, a, feature, by, using, a, maximum, entrophy*
	information, gain, method, method, using, scoring, techniques, weighting, feature, using, maximum, entrophy
D9	*selected, features, is, featured, with, information, gain, values, greater, than, or, equal, to, a, certain, threshold, value*
	selected, features, featured, information, gain, values, greater, equal, certain, threshold, value
D10	*the, threshold, value, used, is, and*
	threshold, value
Q-ID	seleksi, fitur
	seleksi, fitur
Q-EN	feature, selection
	feature, selection

4. *Stemming*

   Token yang sudah difilter dan tidak memiliki *stopword* akan di-*stemming* untuk menemukan kata dasarnya.

Tabel 4.5 Tabel Hasil *Stemming*

No. Dokumen	Isi dokumen
D1	klasifikasi, diterapkan, bidang, kehidupan, termasuk, teks

Tabel 4.5 Tabel Hasil *Stemming* (lanjutan)

No. Dokumen	Isi dokumen
	klasifikasi, terap, bidang, hidup, masuk, teks
D2	algoritma, klasifikasi, menggunakan, fitur, data, membangun, model
	algoritma, klasifikasi, guna, fitur, data, bangun, model
D3	fitur, sesuai, hasil, klasifikasi
	fitur, sesuai, hasil, klasifikasi
D4	seleksi, fitur, teknik, memilih, fitur, penting, relevan, data, mengurangi, fitur, relevan.
	seleksi, fitur, teknik, pilih, fitur, penting, relevan, data, kurang, fitur, relevan
D5	seleksi, fitur, bertujuan, memillih, fitur, terbaik, kumpulan, data, fitur
	seleksi, fitur, tuju, pilih, fitur, baik, kumpul, data, fitur
D6	*feature, selection, aims, select, best, features, dataset, features*
	feature, select, aim, select, best, feature, dataset, feature
D7	*purpose, research, apply, information, gain, method, feature, selection, system, indonesian, language, text, documents*
	purpose, search, apply, information, gain, method, feature, select, system, indonesia*, language, text, document*
D8	*information, gain, method, method, using, scoring, techniques, weighting, feature, using, maximum, entrophy*
	information, gain, method, method, use, score, technique, weight, feature, maximum, enthrophy
D9	*selected, features, featured, information, gain, values, greater, equal, certain, threshold, value*
	select, feature, feature, information, gain, value, great, equal, certain, threshold, value
D10	*threshold, value*
	threshold, value
Q-ID	seleksi, fitur
	seleksi, fitur
Q-EN	feature, selection
	feature, selection

Setelah ditemukan *term* tiap dokumen yang dapat digunakan sebagai fitur, maka dapat dilanjutkan ke proses pembobotan *term*. *Term* penyusun tiap dokumen dituliskan dalam Tabel 4.6.

Tabel 4.6 Tabel Dokumen dan *Term* Penyusunnya

No. Dokumen	*Term*
D1	klasifikasi, terap, bidang, hidup, masuk, teks
D2	algoritma, klasifikasi, guna, fitur, data, bangun, model
D3	fitur, sesuai, hasil, klasifikasi
D4	seleksi, fitur, teknik, pilih, fitur, penting, relevan, data, kurang, fitur, relevan
D5	seleksi, fitur, tuju, pilih, fitur, baik, kumpul, data, fitur
D6	*feature, select, aim, select, best, feature, dataset, feature*
D7	*purpose, search, apply, information, gain, method, feature, select, system,* indonesia*, language, text, document*
D8	*information, gain, method, method, use, score, technique, weight, feature, maximum, enthrophy*
D9	*select, feature, feature, information, gain, value, great, equal, certain, threshold, value*
D10	*threshold, value*
Q-ID	seleksi, fitur
Q-EN	*select, feature*

4.6.2 Text Weighting

Tahap selanjutnya adalah pembobotan teks. Hasil dari setiap subproses yang dilakukan untuk menghitung bobot tiap *term* ditunjukan pada Tabel 4.7 sampai Tabel 4.9.

1. Penghitungan t*f* dan *df*

   Nilai *tf* didapatkan dari menghitung jumlah kemunculan *term* dalam dokumen, sementara *df* didapatkan dengan menghitung berapa banyak dokumen yang mengandung *term* tersebut. Hasil penghitungan *tf* dan *df term* yang didapatkan dari tahap *preprocessing* dituliskan dalam Tabel 4.7.

Tabel 4.7 Tabel Penghitungan *Tf* dan *df*

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN	*df*
klasifikasi	1	1	1										3
terap	1												1
bidang	1												1
hidup	1												1
masuk	1												1
teks	1												1
algoritma		1											1
guna		1											1
fitur		1	1	3	3						1		5
dapat		1											1
data		1		1	1								3

Tabel 4.7 Tabel Penghitungan *Tf* dan *df* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN	*df*
bangun		1											1
model		1											1
sesuai			1										1
hasil			1										1
seleksi				1	1						1		3
teknik				1									1
pilih				1	1								2
penting				1									1
relevan				2									1
hadap				1									1
kurang				1									1
tuju					1								1
baik					1								1
kumpul					1								1
feature						3	1	1	2			1	5
aim						1							1
select						2	1		1			1	4
best						1							1
dataset						1							1
purpose							1						1
search							1						1
apply							1						1
information							1	1	1				3
gain							1	1	1				3
method							1	2					2
system							1						1
indonesia							1						1
language							1						1
text							1						1
document							1						1
use								1					1
score								1					1
technique								1					1
weight								1					1
maximum								1					1
enthrophy								1					1
value									2	1			2
great									1				1
equal									1				1
certain									1				1
threshold									1	1			2

2. Penghitungan *w(tf)* dan *idf*

Nilai *w(tf)* adalah hasil *log* dari nilai *tf*, sementara *idf* adalah hasil *log* dari perbandingan jumlah keseluruhan dokumen dengan nilai *df*. Hasil penghitungan *w(tf)* dan *idf term* dituliskan dalam Tabel 4.8.

Tabel 4.8 Tabel Penghitungan *w(tf)* dan *idf*

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN	*idf*
klasifikasi	1	1	1										0,60
terap	1												1,08
bidang	1												1,08
hidup	1												1,08
masuk	1												1,08
teks	1												1,08
algoritma		1											1,08
guna		1											1,08
fitur		1	1	1,48	1,48						1		0,38
dapat		1											1,08
data		1		1	1								0,60
bangun		1											1,08
model		1											1,08
sesuai			1										1,08
hasil			1										1,08
seleksi				1	1						1		0,60
teknik				1									1,08
pilih				1	1								0,78
penting				1									1,08
relevan				1,30									1,08
hadap				1									1,08
kurang				1									1,08
tuju					1								1,08
baik					1								1,08
kumpul					1								1,08
feature						1,48	1	1	1,30			1	0,38
aim						1							1,08
select						1,30	1		1			1	0,48
best						1							1,08
dataset						1							1,08
purpose							1						1,08
search							1						1,08
apply							1						1,08
information							1	1	1				0,60
gain							1	1	1				0,60
method							1	1,30					0,78

Tabel 4.8 Tabel Penghitungan *w(tf)* dan *idf* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN	*idf*
system							1						1,08
indonesia							1						1,08
language							1						1,08
text							1						1,08
document							1						1,08
use								1					1,08
score								1					1,08
technique								1					1,08
weight								1					1,08
maximum								1					1,08
enthrophy								1					1,08
value									1,30	1			0,78
great									1				1,08
equal									1				1,08
certain									1				1,08
threshold									1	1			0,78

3. Penghitungan *W(t,d)*

   Nilai bobot *term W(t,d)* didapatkan dengan mengalikan nilai *w(tf)* dan nilai *idf*. Hasilnya dituliskan dalam Tabel 4.9.

Tabel 4.9 Tabel Penghitungan *W(t,d)*

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN
klasifikasi	0,60	0,60	0,60									
terap	1,08											
bidang	1,08											
hidup	1,08											
masuk	1,08											
teks	1,08											
algoritma		1,08										
guna		1,08										
fitur		0,38	0,38	0,56	0,56						0,38	
dapat		1,08										
data		0,60		0,60	0,60							
bangun		1,08										
model		1,08										
sesuai			1,08									
hasil			1,08									
seleksi				0,60	0,60						0,60	
teknik				1,08								
pilih				0,78	0,78							
penting				1,08								

Tabel 4.9 Tabel Penghitungan *W(t,d)* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN
relevan				1,40								
hadap				1,08								
kurang				1,08								
tuju					1,08							
baik					1,08							
kumpul					1,08							
feature						0,56	0,38	0,38	0,49			0,38
aim						1,08						
select						0,62	0,48		0,48			0,48
best						1,08						
dataset						1,08						
purpose							1,08					
search							1,08					
apply							1,08					
information							0,60	0,60	0,60			
gain							0,60	0,60	0,60			
method							0,78	1,01				
system							1,08					
indonesia							1,08					
language							1,08					
text							1,08					
document							1,08					
use								1,08				
score								1,08				
technique								1,08				
weight								1,08				
maximum								1,08				
enthrophy								1,08				
value									1,01	0,78		
great									1,08			
equal									1,08			
certain									1,08			
threshold									0,78	0,78		

4.6.3 Vector Space Model

Untuk menemukan kemiripan dokumen dengan *query* digunakan metode *vector space model* (*VSM*). Sebelum menghitung jarak antara dokumen dengan *query*, panjang vektor tiap *term* dalam dokumen perlu dihitung terlebih dahulu. Setelah itu, dihitung nilai *dot product* antara dokumen dengan *query*. Terakhir

adalah penghitungan *cosine similarity* yang didapatkan dari hasil pembagian *dot product* dengan panjang vektor dokumen dan *query*.

1. Panjang vektor dokumen dan *query*

   Panjang vektor didapatkan dengan menguadratkan bobot *W(t,d)* tiap *term* dalam dokumen dan *query*. Hasilnya dituliskan dalam Tabel 4.10

Tabel 4.10 Tabel Panjang Vektor Dokumen dan *Query*

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN
klasifikasi	0,362	0,362	0,362									
terap	1,165											
bidang	1,165											
hidup	1,165											
masuk	1,165											
teks	1,165											
algoritma		1,165										
guna		1,165										
fitur		0,145	0,145	0,315	0,315						0,145	
dapat		1,165										
data		0,362		0,362	0,362							
bangun		1,165										
model		1,165										
sesuai			1,165									
hasil			1,165									
seleksi				0,362	0,362						0,362	
teknik				1,165								
pilih				0,606	0,606							
penting				1,165								
relevan				1,971								
hadap				1,165								
kurang				1,165								
tuju					1,165							
baik					1,165							
kumpul					1,165							
feature						0,315	0,145	0,145	0,245			0,145
aim						1,165						
select						0,385	0,228		0,228			0,228
best						1,165						
dataset						1,165						
purpose							1,165					
search							1,165					
apply							1,165					
information							0,362	0,362	0,362			
gain							0,362	0,362	0,362			

Tabel 4.10 Tabel Panjang Vektor Dokumen dan *Query* (lanjutan)

Term	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	QID	QEN
method							0,606	1,025				
system							1,165					
indonesia							1,165					
language							1,165					
text							1,165					
document							1,165					
use								1,165				
score								1,165				
technique								1,165				
weight								1,165				
maximum								1,165				
enthrophy								1,165				
value									1,025	0,606		
great									1,165			
equal									1,165			
certain									1,165			
threshold									0,606	0,606		
Jumlah	6,186	6,693	2,836	8,276	5,140	4,195	11,020	8,882	6,322	1,211	0,507	0,372
SQRT (Jumlah)	2,487	2,587	1,684	2,877	2,267	2,048	3,320	2,980	2,514	1,100	0,712	0,610

2. *Dot Product* dokumen dengan *query*

   Dot product didapatkan dengan mengalikan bobot tiap *term* dalam dokumen dengan bobot *query*. Secara otomatis, *term* yang tidak ada dalam *query* akan bernilai 0. Hasilnya dituliskan pada Tabel 4.11 untuk *query* berbahasa Indonesia *(Q-ID)* dan pada Tabel 4.12 untuk *query* bahasa Inggris *(Q-EN)*.

Tabel 4.11 Tabel *Dot Product* Dokumen dengan *Query* Bahasa Indonesia

Term	D2	D3	D4	D5
fitur	0,144	0,144	0,213	0,213
seleksi			0,36	0,36
Jumlah	0,144	0,144	0,573	0,573

Tabel 4.12 Tabel *Dot Product* Dokumen dengan *Query* Bahasa Inggris

Term	D6	D7	D8	D9
feature	0,213	0,144	0,144	0,186
select	0,298	0,23		0,23
Jumlah	0,51	0,375	0,144	0,417

3. *Cosine Similarity* dokumen dengan *query*
   Nilai *similarity* dihitung dengan menggunakan Persamaan 2.2. Sebagai contoh, untuk mendapatkan nilai *similarity* antara dokumen 2 (*D2*) dengan *query* berbahasa Indonesia (*Q-ID*) dapat dihitung sebagai berikut:

$$sim\ (\overline{D}_2, \overline{Q_{ID}}) = \frac{\overline{D} \cdot \overline{Q}}{|\overline{D}| \times |\overline{Q}|}$$
$$sim\ (\overline{D}_2, \overline{Q_{ID}}) = \frac{0{,}144}{2{,}587 \times 0{,}7102}$$
$$sim\ (\overline{D}_2, \overline{Q_{ID}}) = 0{,}0781$$

   Hasil *cosine similarity* disusun dengan cara mengurutkan dokumen dari yang *similarity*-nya paling tinggi. Hasilnya, D6 akan ditampilkan pertama dengan nilai *similarity* sebesar 0,407 terhadap *query* dan D2 akan ditampilkan terakhir, sesuai dengan Tabel 4.13 berikut.

Tabel 4.13 Tabel Hasil Penghitungan *Cosine Similarity*

	cosim	Relevan/Tidak Relevan
D6	0,407	Relevan
D5	0,355	Relevan
D4	0,28	Relevan
D9	0,27	Relevan
D7	0,184	Relevan
D3	0,12	Tidak Relevan
D8	0,079	Tidak Relevan
D2	0,078	Tidak Relevan

4.6.4 Pengujian dengan *P@K*

Hasil pengujian manualisasi dapat dihitung menggunakan Persamaan 2.6. Hasilnya secara lengkap dituliskan dalam Tabel 4.14.

Tabel 4.14 Tabel Hasil Pengujian *Precision at 8*

K	Daftar dokumen *top-8*	Jumlah dokumen relevan dalam *top-8*	*P @ 8*
8	D6 (Relevan) D5 (Relevan) D4 (Relevan) D9 (Relevan) D7 (Relevan) D3 (Tidak Relevan) D8 (Tidak Relevan) D2 (Tidak Relevan)	8	$\frac{5}{8} = 0{,}625$

4.6.5 Pengujian dengan *PRC*

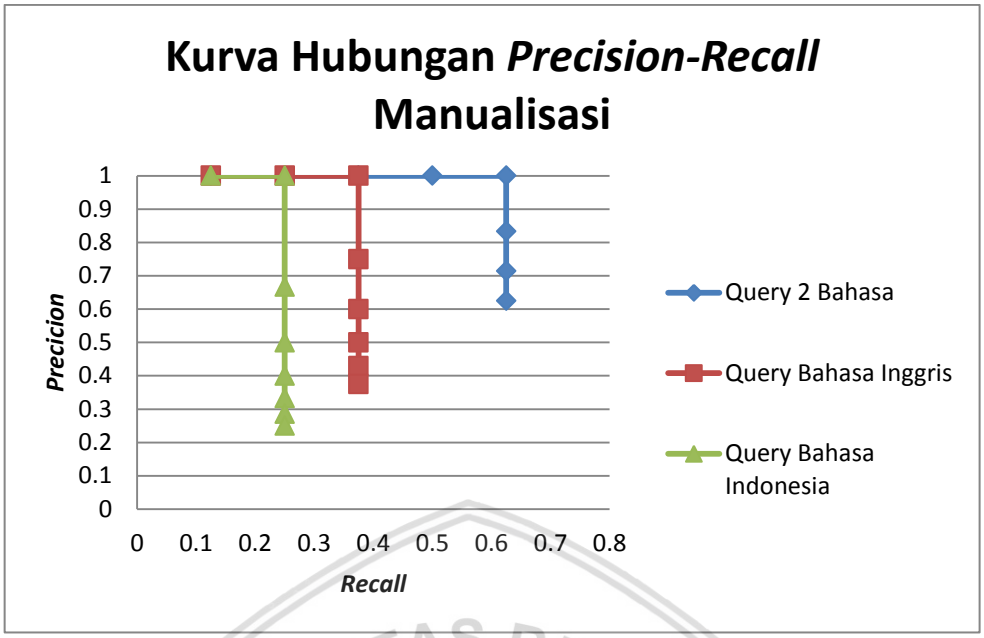
Nilai *precision* dan *recall* dapat dihitung dengan menggunakan Persamaan 2.7 dan Persamaan 2.8. Dari hasil *cosine similarity* pada Tabel 4.13, dapat ditentukan nilai *precision* dan *recall* untuk tiap tingkatan dokumen. Hasilnya tertera pada Tabel 4.15. Nilai hasil pengujian untuk *query* yang hanya menggunakan 1 bahasa pada Tabel 4.16 digunakan sebagai pembanding nilai *precision* dan *recall* untuk Tabel 4.15. Perbandingannya digambarkan dalam bentuk kurva pada Gambar 4.9.

Tabel 4.15 Tabel Hasil Pengujian *Precision* dan *Recall* dengan *Query* 2 Bahasa

Menggunakan *query* Berbahasa Inggris dan Indonesia				
No.	Dokumen	Relevansi	*Recall*	*Precision*
1.	D6	Relevan	0,125	1
2.	D5	Relevan	0,25	1
3.	D4	Relevan	0,375	1
4.	D9	Relevan	0,5	1
5.	D7	Relevan	0,625	1
6.	D3	Tidak Relevan	0,625	0,833
7.	D8	Tidak Relevan	0,625	0,714
8.	D2	Tidak Relevan	0,625	0,625
		Rata-rata:	0,469	0,897

Tabel 4.16 Tabel Hasil Pengujian *Precision* dan *Recall* dengan *Query* 1 Bahasa

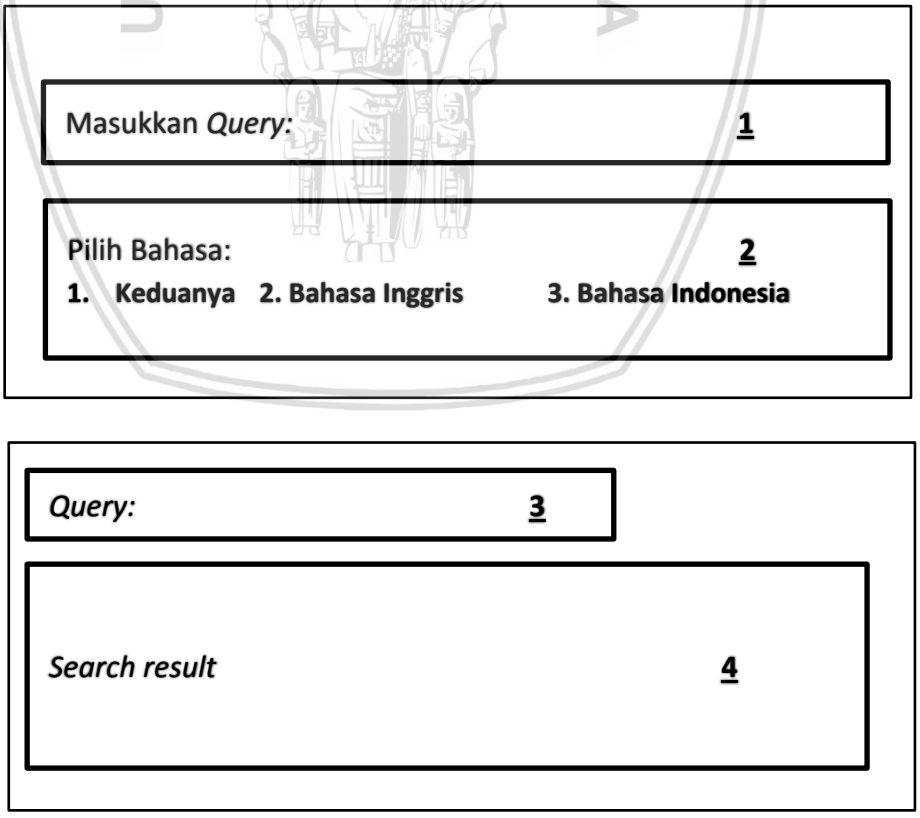
Menggunakan *query* Berbahasa Inggris					Menggunakan *query* Berbahasa Indonesia			
No.	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*
1.	D6	Relevan	0,125	1	D5	Relevan	0,125	1
2.	D9	Relevan	0,25	1	D4	Relevan	0,25	1
3.	D7	Relevan	0,375	1	D3	Tidak Relevan	0,25	0,667
4.	D8	Tidak Relevan	0,375	0,75	D2	Tidak Relevan	0,25	0,5
5.	D2	Tidak Relevan	0,375	0,6	D6	Tidak Relevan	0,25	0,4
6.	D3	Tidak Relevan	0,375	0,5	D7	Tidak Relevan	0,25	0,333
7.	D4	Tidak Relevan	0,375	0,429	D8	Tidak Relevan	0,25	0,286
8.	D5	Tidak Relevan	0,375	0,375	D9	Tidak Relevan	0,25	0,25
		Rata-rata:	0,328	0,707			0,234	0,554



Gambar 4.9 Kurva Perbandingan *Precision-Recall* Manualisasi

4.7 Perancangan Antarmuka Sistem

Perancangan antarmuka sistem pencarian jurnal ilmiah dengan menggunakan metode VSM seperti yang tertera pada Gambar 4.10.



Gambar 4.10 Perancangan Antarmuka Sistem

Keterangan:

1. Pengguna memasukkan *query* untuk melakukan pencarian jurnal ilmiah. *Query* yang dimasukkan dapat berupa bahasa Indonesia atau bahasa Inggris, diikuti tombol *enter.*
2. Pengguna dapat memilih bahasa jurnal yang ingin dicari, tombol 1 untuk mencari jurnal dalam bahasa Indonesia dan Inggris, tombol 2 untuk mencari jurnal dalam bahasa Inggris saja, dan tombol 3 untuk mencari jurnal dalam bahasa Indonesia saja, diikuti tombol *enter*.
3. *Query* yang dimasukkan pengguna beserta hasil terjemahan *query* ditampilkan.
4. Hasil pencarian jurnal ditampilkan dibawah *query*. Hasil yang ditampilkan adalah nilai kemiripan *query* dengan dokumen (*cosine similarity*-nya), *path folder* dokumen, judul, penulis, dan abstrak jurnal.

4.8 Perancangan Pengujian

Dalam penelitian ini akan digunakan 2 jenis pengujian, yakni pengujian dengan *precision @ K* untuk menghitung akurasi hasil temu kembali oleh sistem dan mambandingkan nilai *precision-recall curve* untuk melihat pengaruh penerjemahan *query* terhadap hasil temu kembali.

4.8.1 Pengujian Akurasi Hasil Temu Kembali dengan *Precision @ K*

Precision @ K menghitung nilai *precision* pada *top-k* dokumen. Jenis pengujian ini cocok digunakan untuk hasil temu kembali yang berperingkat karena melihat urutan dokumen yang ditemukan kembali. Perancangan pengujian akurasi ini dituliskan dalam Tabel 4.17.

Tabel 4.17 Tabel Pengujian dengan *Precision @ K*

Query	Daftar dokumen *top-k*	Jumlah dokumen relevan dalam *top-k*	*P@K*
Q1			

4.8.2 Pengujian Pengaruh Penerjemahan *Query* dengan *Precision-Recall Curve*

Pengujian ini dilakukan untuk menemukan pengaruh penerjemahan *query* terhadap hasil temu kembali dengan mengukur nilai *precision* dan *recall*-nya. Tabel 4.18 adalah perancangan pengujian untuk menghitung nilai *precision* dan *recall* dibeberapa tingkatan dokumen yang ditemukan kembali.

Tabel 4.18 Tabel Pengujian dengan *PRC*

Dokumen	Relevansi	*Recall*	*Precision*

4.9 Implementasi Program

Implementasi program dilakukan dengan mengacu pada hasil perancangan *flowchart* yang telah dilakukan pada subbab sebelumnya. Bahasa pemrograman yang digunakan dalam implementasi program ini adalah *Python* dengan IDE *Spyder*. Spesifikasi *hardware* dan *software* dijelaskan pada subbab analisis kebutuhan sistem. Implementasi program terdiri dari 4 bagian, antara lain tahap penerjemahan *query*, *preprocessing*, pembobotan teks, dan pengukuran kemiripan dokumen dengan *query*.

4.9.1 Implementasi Penerjemahan *Query*

Proses pencarian dimulai dengan menerjemahkan *query* yang dimasukkan pengguna. Pada tahap ini digunakan *library Googletrans* yang mengimplementasikan *Google Translate API* untuk membantu proses terjemahan. Dari *query* yang dimasukkan pengguna akan dideteksi bahasanya dengan menggunakan *library langdetect*. Kode program untuk penerjemahan *query* dituliskan pada Kode Program 4.1.

```
1   def detectLanguage (words):
2       w = words.split()
3       base = 'E:/Fixed Data/base_en.txt'
4       openBase = open(base, "r")
5       readEN = []
6       for word in openBase.readlines():
7           readEN.append(str(word).strip('\n'))
8       language = detect_langs (words)
9       for listLang in language:
10          for wr in w:
11              if listLang is 'en' or wr in readEN :
12                  return 'en'
13              else:
14                  return 'id'

15  def translate(query):
16      query2 = []
17      qSplit = query.split()
18      for words in qSplit:
19          if detectLanguage(words) == "en":
20              tempQ2 = translator.translate(words, dest='id', src
    ='en').text
21              query2.append(tempQ2)
22          elif detectLanguage(words) == "id":
23              tempQ2 = translator.translate(words, dest='en', src
    = 'id').text
24              query2.append(tempQ2)
25      query2 = " ".join(query2)
26      return query2
```

Kode Program 4.1 Penerjemahan *Query*

Keterangan:

1 Membuat *method detectLanguage* untuk mendeteksi bahasa. Keluaran dari *method* ini adalah jenis bahasa yang digunakan pengguna, antara bahasa Inggris (*'en'*) atau bahasa Indonesia (*'id'*)

2 Menginisialisasi variabel *w* dengan *list* yang berisi kata-kata dari masukan pengguna

3 Menginisialisasi variabel *base* yang akan digunakan sebagai kamus

	bantuan untuk mendeteksi bahasa
4	Menginisialisasi variabel *openBase* untuk membuka *file* yang ada pada variabel *base*
5	Menginisialisasi variabel *readEN* dengan *list* kosong
6	Melakukan perulangan untuk setiap indeks yang ada pada *list openBase*
7	Setiap kata yang ada pada *list openBase* akan ditambahkan ke dalam variabel *readEN*
8	Menginisialisasi variabel *language* dengan objek *detect_langs*
9	Melakukan perulangan untuk setiap indeks yang ada pada *list language*
10	Melakukan perulangan untuk setiap indeks yang ada pada *list w*
11	Kondisi jika variabel *listLang* bernilai *'en'* atau indeks *wr* ada di dalam
12	variabel *readEN* dipenuhi, maka sistem akan mengembalikan nilai *'en'*
13-	Jika kondisi sebelumnya tidak dipenuhi, maka *method* mengembalikan
14	nilai *'id'*
15	Membuat *method translate* yang digunakan untuk menerjemahkan *query* masukan pengguna
16	Menginisialisasi variabel query2 dengan *list* kosong
17	Menginisialisasi variabel *qSplit* dengan *query* masukan pengguna yang dipisah per kata
18	Melakukan perulangan untuk setiap indeks yang ada pada *list* qSplit
19	Kondisi jika *method detectLanguage* untuk variabel *words* mengembalikan
20	nilai *'en',* variabel *tempQ2* akan memanggil *method translate* dari objek *translator* untuk melakukan penerjemahan dari bahasa Inggris ke bahasa Indonesia
21	Menambahkan variabel *tempQ2* ke dalam *list query2*
22-	Kondisi jika method *detectLanguage* untuk variabel *words* mengembalikan nilai *'id'*, variabel *tempQ2* akan memanggil *method*
23	*translate* dari objek *translator* untuk melakukan penerjemahan dari bahasa Indonesia ke bahasa Inggris
24	Menambahkan variabel *tempQ2* ke dalam *list query2*
25	Menggabungkan *list query2* menjadi satu *string*
26	*Method* mengembalikan hasil penerjemahan yaitu *query2*

4.9.2 Implementasi *Text Preprocessing*

Sebelum dihitung bobotnya, *query* dan dokumen harus melalui tahap *preprocessing* untuk menghilangkan simbol-simbol dan kata-kata yang tidak penting. Ada 4 tahapan *preprocessing* yang harus dilakukan, yang pertama adalah *case folding* yang memanfaatkan *regular expression (regex)* untuk menyeragamkan teks. Langkah kedua adalah *tokenizing* atau tokenisasi. Pada tahap ini digunakan *library* dari *NLTK* untuk memecah *string* menjadi token-token yang disimpan dalam satu *list*. Langkah ketiga adalah *stopword removal.* Daftar *stopwords* yang digunakan diambil dari *library NLTK*. Masukan yang akan diproses ditahap ini adalah token-token dari proses tokenisasidan keluarannya adalah *list term* yang sudah tidak memiliki *stopword*. Langkah terakhir adalah *stemming.* Dalam program, digunakan *library Sastrawi* yang

mengimplementasikan algoritme *stemming* Nazief dan Adriani dan *NLTK* yang mengimplementasikan algoritme *stemming* Porter. Hasilnya berupa *list term* untuk diproses ditahap selanjutnya.

```
1  def caseFold (words):
2      words = regex.sub(' ', str.lower(words)).strip()
3      return words

4  def tokenizing (words):
5      words = nltk.word_tokenize(words) #tokenizing
6      return words

7  def stopwordRem(words):
8      listTerms = []
9      for term in words:
10         if term not in stopwords.words('indonesian') and term
   not in stopwords.words('english'):
11             listTerms.append(term)
12     return listTerms

13 def stemming(words):
14     listTerms = []
15     tempWL = None
16     tempWL = " ".join(words)
17     for term in words:
18         if detectLanguage(tempWL)=="id":
19             stem = stemmer.stem(str(term))
20             listTerms.append(stem)
21         elif detectLanguage(tempWL)=="en":
22             stem2 = stemEN.stem(term)
23             listTerms.append(stem2)
24     return listTerms
```

Kode Program 4.2 Fungsi *Text Preprocessing*

Keterangan:

1	Membuat *method caseFold* untuk mengubah seluruh huruf kapital menjadi huruf kecil
2	Menginisialisasi variabel *words* dengan masukan yang sudah dijadikan huruf kecil dengan menggunakan *regex*
3	*Method* mengembalikan nilai variabel *words*
4	Membuat method *tokenizing* untuk memecah *string* menjadi *term*
5	Menginisialisasi variabel *words* dengan memanggil fungsi *word_tokenize* dari *library nltk* untuk mengimplementasikan tahap tokenisasi
6	*Method* mengembalikan nilai variabel *words*
7	Membuat *method stopwordRem* untuk menghilangkan *stopword*
8	Menginisialisasi variabel *listTerms* dengan *list* kosong
9	Melakukan perulangan untuk tiap indeks pada *list words*
10-11	Kondisi jika *term* tidak ada dalam *list stopword* berbahasa Indonesia dan bahasa Inggris, maka *term* akan ditambahkan ke dalam *list listTerms*
12	*Method* mengembalikan nilai variabel *listTerms*
13	Membuat *method stemming* untuk mengimplementasikan tahapan *stemming*
14	Menginisialisasi variabel *listTerms* dengan *list* kosong
15	Menginisialisasi variabel *tempWL* dengan nilai *none*

16 Menggabungkan *list words* menjadi satu *string* dan disimpan dalam variabel *tempWL*

17 Melakukan perulangan untuk tiap indeks pada *list words*

18-19 Jika *method detectLanguage* dengan masukan *tempWL* bernilai *'id'* maka akan dilakukan stemming dengan memanggil fungsi *stem* dari *library Sastrawi*

20 Hasil *stemming* ditambahkan ke dalam *list listTerms*

21-22 Jika kondisi sebelumnya tidak dipenuhi dan *method detectLanguage* mengembalikan nilai *'en',* akan dilakukan *stemming* dengan memanggil fungsi *stem* dari *library NLTK*

23 Hasil *stemming* ditambahkan ke dalam *list listTerms*

24 *Method* mengembalikan nilai variabel listTerms

4.9.3 Implementasi Pembobotan Teks (*Text Weighting)*

Nilai pembobotan *term* didapat dengan mengalikan nilai logaritma jumlah *term* dengan invers frekuensi dokumen, seperti pada Persamaan 2.3. Hasil penghitungan *wtf, idf,* dan *W(t,d)* disimpan dalam bentuk *dictionary* yang memiliki pasangan *keys* (direpresentasikan oleh *term)* dan *values*.

```
1  def wtf(dokumen):
2      termDict = {}
3      wtfDict = {}
4      termFreq = []
5      for terms in dokumen:
6          termDict[terms] = termDict.get(terms, 0)+1
7      termF = termDict.values()
8      for freq in termF:
9          newWeight=math.log10(freq)+1
10         termFreq.append(newWeight)
11     wtfDict = combinedDict(termDict.keys(), termFreq)
12     return wtfDict

13 def idf(listFile):
14     dumpList = []
15     dfDict = {}
16     idfDict = {}
17     for dok in listFile:
18         termFr = dok.keys()
19         dumpList.extend(termFr)
20     for term in dumpList:
21         dfDict[term] = dfDict.get(term,0)+1
22     totalDoc = len(listFile)
23     oldList = dfDict.values()
24     idf =[]
25     for dfD in oldList:
26         inverseDf = math.log10(totalDoc/dfD)
27         idf.append(inverseDf)
28     idfDict = combinedDict(dfDict.keys(), idf)
29     return idfDict

30 def wtd(listTf, dictDf):
31     for dok in listTf:
32         for key in dictDf:
33             if key in dok:
34                 temp = dok[key]*dictDf[key]
35                 dok[key] = temp
```

```
36      return listTf
```

Kode Program 4.3 Fungsi *Text Weighting*

Keterangan:

1 Membuat *method wtf* untuk menghitung nilai logaritma *tf*

2-4 Menginisialisasi variabel *termDict, wtfDict,* dan *termFreq*

5-6 Perulangan untuk menghitung frekuensi *term*. Jika program menemukan kata yang sama, *value* untuk *term* tersebut akan bertambah 1

7 Menginisialisasi variabel *termF* dengan list *value* dari *dictionary termDict*

8-9 Perulangan untuk menghitung nilai logaritma frekuensi dan hasilnya disimpan pada *variable list* yang baru. Hasilnya akan mengembalikan sebuah *variable* bertipedata *dictionary* dengan pasangan *keys values*-nya adalah *terms* dengan nilai *wtf*-nya.

10 Menambahkan nilai variabel *newWeight* ke dalam *list termFreq*

11 Menginisialisasi variabel *wtfDict* dengan menggabungkan *keys* dari *dictionary termDict* dan *list termFreq*

12 *Method* mengembalikan nilai variabel *wtfDict*

13 Membuat method *idf* untuk menghitung nilai invers frekuensi dokumen

14-16 Menginisialisasi variabel *dumpList, dfDict,* dan *idfDict*

17-18 Perulangan untuk mengumpulkan seluruh *keys* yang diperolah pada tahap penghitungan *wtf*.

19 Menambahkan nilai variabel *termFr* ke dalam *list dumpList*

20-21 Perulangan untuk menghitung jumlah *term*. Hasilnya adalah jumlah kemunculan *term* dari keseluruhan jumlah dokumen yang ada.

22 Inisialisasi variabel *totalDoc* yang berisi panjang variabel *listFile*

23 Inisialisasi variabel *oldList* dengan *list values* milik *dfDict*

24 Inisialisasi variabel *idf*

25-26 Perulangan untuk menghitung *idf.* Nilai invers didapatkan sesuai dengan Persamaan 2.2, yaitu nilai logaritma dari pembagian total dokumen (variabel *totalDoc*) dengan jumlah kemunculan *term* dalam keseluruhan dokumen (tiap nilai dalam *oldList)*

27 Menambahkan nilai variabel *inverseDf* ke dalam *list idf*

28 Menginisialisasi variabel *idfDict* yang berisi *keys* variabel *dfDict* dan nilai *idf*-nya

29 *Method idf* mengembalikan nilai variabel *idfDict*

30 Membuat *method wtd* untuk mengimplementasikan penghitungan *TF.IDF* sesuai dengan Persamaan 2.3.

31 Melakukan perulangan untuk mendapatkan *dok* dalam variabel *listTf*

32 Melakukan perulangan bersarang untuk mendapatkan *key* dalam variabel *dictDf*

33-34 Melakukan pencocokan *key* dengan *dok*. Jika cocok, maka nilai *wtf* (*dok[key]*) dikalikan dengan nilai idf (*dictDf[key]*).

35 Nilai *dok[key]* di-*update* dengan hasil kali *wtf* dan *idf*

36 *Method wtd* mengembalikan nilai variabel *listTf*

Pada penghitungan bobot untuk dokumen jurnal ilmiah, *terms* dan nilai *W(t,d)* disimpan dalam sebuah *file* berekstensi *.CSV* untuk mempercepat waktu pencarian.

4.9.4 Implementasi *Vector Space Model*

Penghitungan kemiripan vektor dokumen dan *query* dalam penelitian ini menggunakan nilai *cosine similarity*, yang secara matematis dituliskan dalam Persamaan 2.4.

```
def dotProduct(wtdQuery, wtdDok):
    copywtdDok = wtdDok [:]
    qDict = {} #dictionary query
    dProd = []
    for qList in wtdQuery:
        qDict.update(qList)
    for dokList in copywtdDok:
        for key in dokList:
            if key in qDict:
                dpr = qDict[key]*dokList[key]
                dokList[key] = dpr
            else:
                dokList[key]=0
    for item in copywtdDok:
        value = item.values()
        value = sum(value)
        dProd.append(value)
    return dProd
def vectorLength(document):
    vLength = []
    for item in document:
        vals = item.values()
        vals2 = [x**2 for x in vals]
        vals2 = sum(vals2)
        temp = math.sqrt(vals2)
        vLength.append(temp)
    return vLength
inBhs = raw_input ("1. English\n2. Bahasa Indonesia\n3. Both\nChoose Language: ")
def vsm (qLength, dokLength, dotPrd):
    vsmList = []
    if (inBhs == '1'):
        for i in qLength:
            for index, j in enumerate (dokLength, 1):
                for indeks, k in enumerate (dotPrd, 1):
                    if (j>0 and i>0 and index == indeks):
                        if (index <= 500):
                            pathTemp = 'E:\Fixed Data\dok%s.txt'%index
                            tempRead = open(pathTemp, "r")
                            tRead = tempRead.readlines()
                            title = tRead[1].strip('\n')
                            author = tRead [0].strip('\n')
                            abstract = tRead [2]
                            abstract = abstract[:250]+'...'
                            tempRead.close()
                            cosim = k/(j*i)
                            vsm = (cosim, pathTemp, title, author, abstract)
                            if cosim > 0 :
                                vsmList.append(vsm)
                            else:
                                cosim = 0
    elif (inBhs == '2'):
        #index = [501:1000]
        for i in qLength:
            for index, j in enumerate (dokLength, 1):
                for indeks, k in enumerate (dotPrd, 1):
                    if (j>0 and i>0 and index == indeks):
```

```
57                          if (index >= 501):
58                              pathTemp = 'E:\Fixed Data\dok%s.txt'%index
59                              tempRead = open(pathTemp, "r")
60                              tRead = tempRead.readlines()
61                              title = tRead[1].strip('\n')
62                              author = tRead [0].strip('\n')
63                              abstract = tRead [2]
64                              abstract = abstract[:250]+'...'
65                              tempRead.close()
66                              cosim = k/(j*i)
67                              vsm = (cosim, pathTemp, title, author,
abstract)
68                              if cosim > 0 :
69                                  vsmList.append(vsm)
70                              else:
71                                  cosim = 0
72      elif (inBhs == '3'):
73          for i in qLength:
74              for index, j in enumerate (dokLength, 1):
75                  for indeks, k in enumerate (dotPrd, 1):
76                      if (j>0 and i>0 and index == indeks):
77                          pathTemp = 'E:\Fixed Data\dok%s.txt'%index
78                          tempRead = open(pathTemp, "r")
79                          tRead = tempRead.readlines()
80                          title = tRead[1].strip('\n')
81                          author = tRead [0].strip('\n')
82                          abstract = tRead [2]
83                          abstract = abstract[:250]+'...'
84                          tempRead.close()
85                          cosim = k/(j*i)
86                          vsm = (cosim, pathTemp, title, author, abstract)
87                          if cosim > 0 :
88                              vsmList.append(vsm)
89                          else:
90                              cosim = 0
91      else:
92          print "Input Pilihan Bahasa 1-3!"
93      vsmList = list(set(vsmList))
94      sort = sorted(vsmList, key=lambda tup:tup[0], reverse=True)
95      for retrieved in sort[0:20]:
96          pprint (retrieved)
97      length = len(sort)
98      print "\nTotal Document Retrieved: "+str(length)
```

Kode Program 4.4 Fungsi *Vector Space Model*

Keterangan:

1 Tiap dokumen dan *query* akan dihitung nilai *dotProduct*-nya dengan implementasi pada baris ke-1 sampai 18.
2-4 Menginisialisasi variabel *copywtdDok, qDict,* dan *qProd*
5 Melakukan perulangan untuk tiap nilai dalam *list wtdQuery*
6 Memasukkan nilai *qList* ke dalam *qDict*
7 Perulangan untuk setiap nilai dalam *list copywtdDok*
8 Perulangan untuk setiap nilai dalam *dokList*
9 Syarat kondisi 1: jika variabel *key* = *qDict*, maka variabel *dpr* di-*update* dengan
10 mengalikan nilai *qDict[key]* dan *dokList[key]*
11 Menginisialisasi variabel *dokList[key]* dengan nilai variabel *dpr*
12 Syarat kondisi 2: jika tidak memenuhi kondisi 1 variable *dokList[key]* diinisialisasi
13 dengan nilai 0
14 Perulangan untuk setiap nilai dalam variabel *copywtdDok*

15 Inisialisasi nilai variabel *value* dengan nilai *list value* dari *dictionary item*
16 Inisialisasi nilai variabel *value* dengan jumlah seluruh nilai *value*
17 Menambahkan nilai variabel *value* ke dalam *dProd*
18 *Method dotProduct* mengembalikan nilai *dProd*
19 Membuat method *vectorLength* untuk menghitung panjang vektor dokumen dengan *query*
20 Menginisialisasi variabel *vLength* berupa *list* kosong
21 Perulangan untuk setiap nilai dalam variabel *dokumen*
22 Menginisialisasi nilai *vals* dengan *list values* dari *dictionary item*
23 Inisialisasi variabel *vals2 dengan* dengan nilai *vals* kuadrat
24 Inisialisasi nilai variabel *vals2* dengan jumlah seluruh nilai *vals2*
25 Inisialisasi nilai variabel *temp* dengan nilai akar dari vals2
26 Menambahkan nilai variabel *temp* ke dalam variabel *vLength*
27 *Method vectorLength* mengembalikan nilai *vLength*
28 Mencetak opsi menu pilihan bahasa jurnal ilmiah yang akan ditampilkan, pengguna dapat memilih 1, 2, atau 3
29 Membuat *method vsm* untuk menghitung nilai kemiripan dokumen dengan *query*
30 Menginisialisasi variabel *vsmList*
31-90 Kode pada baris 31-90 dijalankan tergantung bahasa pilihan pengguna sesuai baris ke-28. Secara garis besar, kode pada baris 31-90 mirip, yang membedakan adalah pilihan dokumen yang ingin ditemukan kembali
32 Melakukan perulangan untuk tiap nilai dalam *list qLength*
33 Melakukan perulangan untuk tiap nilai dalam *dokLength*
34 Melakukan perulangan untuk tiap nilai dalam *dotPrd*
35 Kondisi jika nilai *j* dan nilai *I* lebih dari 0, dan nilai *index* sama dengan *indeks* dipenuhi, maka sistem akan melanjutkan eksekusi baris selanjutnya
36 Kondisi jika nilai *index* kurang dari atau sama dengan 500 maka sistem akan melanjutkan eksekusi baris selanjutnya. Dokumen dengan nilai indeks dibawah 501 adalah dokumen berbahasa Inggris
37 Menginisialisasi variabel *pathTemp* dengan *file* dokumen yang akan ditemukan kembali
38 Menginisialisasi variabel *tempRead* untuk membuka *file* yang sudah diinisialisasi pada baris 39
39 Menginisialisasi variabel *tRead* untuk membaca *file* yang sudah dibuka
40 Menginisialisasi variabel *title* untuk mendapatkan judul jurnal ilmiah
41 Menginisialisasi variabel *author* untuk mendapatkan penulis jurnal ilmiah
42 Menginisialisasi variabel *abstract* untuk mendapatkan abstrak jurnal ilmiah
43 Menginisialisasi variabel *abstract* untuk memotong abstrak pada karakter ke-251
44 Menutup *file* yang dibuka *tempRead*
45 Menginisialisasi variabel *cosim*
46 Menginisialisasi variabel *tuple vsm* yang berisi variabel *cosim, pathTemp, title, author, dan abstract*
47-48 Melakukan kondisi jika nilai *cosim > 0* maka *tuple vsm* akan ditambahkan ke dalam *list vsmList*
49 Melakukan kondisi jika kondisi sebelumnya tidak dipenuhi, nilai variabel *cosim* = 0

91 Kondisi jika pengguna tidak memasukan pilihan antara 1 sampai 3, sistem akan
92 memberikan peringatan untuk memilih menu antara 1 sampai 3
93 Menginsialisasi variabel *vsmList*
94 Menginisialisasi variabel *sort* untuk mengurutkan *list vsmList* berdasarkan *cosim* tertinggi
95 Melakukan perulangan untuk menampilkan hasil temu kembali sebanyak 20 dokumen dengan *cosim* tertinggi terhadap *query*
96 Mencetak hasil temu kembali
97 Menginisialisasi variabel *length* untuk menghitung jumlah dokumen yang dianggap relevan oleh sistem secara keseluruhan
98 Mencetak hasil variabel *length*

4.10 Implementasi Antarmuka Pengguna

Implementasi antarmuka pengguna dalam sistem pencarian ini masih menggunakan CLI, sesuai dengan perancangan pada subbab sebelumnya. Contoh hasil implementasi antarmuka pengguna terdapat pada Gambar 4.11.

```
1. English
2. Bahasa Indonesia
3. Both
Choose Language: 3

Input Search Query: smart home system
Translated Query: pintar rumah sistem
==========SEARCHING==========
(0.2655640226916326,
 'E:\\Fixed Data\\dok52.txt',
 'Hardware Design of Smart Home System based on zigBee Wireless Sensor Network,',
 'Zhen-ya Liu,',
 'Abstract: In this paper, the design of the smart home system based on ZigBee technology and GSM / GPRS
network, the smart home system the hardware circuit, using the CC2430 ZigBee wireless sensor networks,
real-time acquisition of the parameters of t...')
(0.24344763637451308,
 'E:\\Fixed Data\\dok838.txt',
 'Kontrol Lampu Berbasis Voice Command Pada Raspberry PI',
 'Muhammad Irfan Reza',
 'Abstract: Smart home merupakan sebuah rumah yang menyediakan kenyamanan, keamanan dan kemudahan kepada
penghuninya dalam pengoperasian setiap peralatan di setiap waktunya. Smart home biasanya terdiri dari
beberapa peralatan elektronik seperti lampu, ...')
(0.23282125591984287,
 'E:\\Fixed Data\\dok801.txt',
 'Implementasi Perangkat Rumah Cerdas Pervasif Menggunakan Komunikasi Websocket dan Struktur Data JSON',
 'Ahmad Mustafidul Ibad',
 'Abstract: Perkembangan teknologi telah memicu pergeseran mendasar dalam gaya hidup modern. Salah satu
penerapan dalam teknologi yang saat ini banyak dikupas secara mendalam adalah rumah cerdas (smart home).
Perangkat dari teknologi rumah cerdas sanga...')
(0.2119923893497516,
 'E:\\Fixed Data\\dok644.txt',
 'Implementasi Fitur Pencahayaan Terjadwal dan Timer pada Smart Light Bulb Pervasive Berbasis ESP8266
Sebagai Home Assistant Menggunakan Aplikasi Berbasis Android',
 'Vatikan Aulia Makkah',
 'Abstract: Internet of Things (IoT) telah mengubah tren masyarakat dunia akan kebutuhan berkomunikasi.
Teknologi IoT mengubah perangkat menjadi sesuatu yang lebih berharga. Konsep Smart Home merupakan salah
satu perkembangan teknologi pada era modern ...')
```

Gambar 4.11 Implementasi Antarmuka Pengguna

BAB 5 HASIL DAN PEMBAHASAN

5.1 Evaluasi dengan *Precision @ K*

Precision @ K digunakan untuk menghitung akurasi sistem. Pengujian dilakukan dengan menentukan apakah *query* yang disediakan relevan terhadap dokumen yang ditemukan kembali sebanyak *top-K*, kemudian dihitung akurasinya dengan Persamaan 2.6. Jumlah dokumen yang akan dicocokan ada 1000 dokumen dengan pembagian sebanyak 500 dokumen berbahasa Inggris dan 500 dokumen berbahasa Indonesia. Jumlah *query* yang akan dicocokan ada 10 *query.* 5 *query* dimasukkan dalam Bahasa Inggris, 5 *query* dimasukkan dalam Bahasa Indonesia. Tiap *query* akan diterjemahkan secara otomatis, *query* berbahasa Inggris akan diterjemahkan ke dalam Bahasa Indonesia, berlaku sebaliknya. Daftar *query* yang akan diuji tertera pada Tabel 5.1 dan daftar dokumennya terlampir di Lampiran A. *Top-K* yang akan diuji untuk tiap *query* adalah *top-5*, *top-10, top-15,* dan *top-20.*

Tabel 5.1 Tabel Daftar *Query*

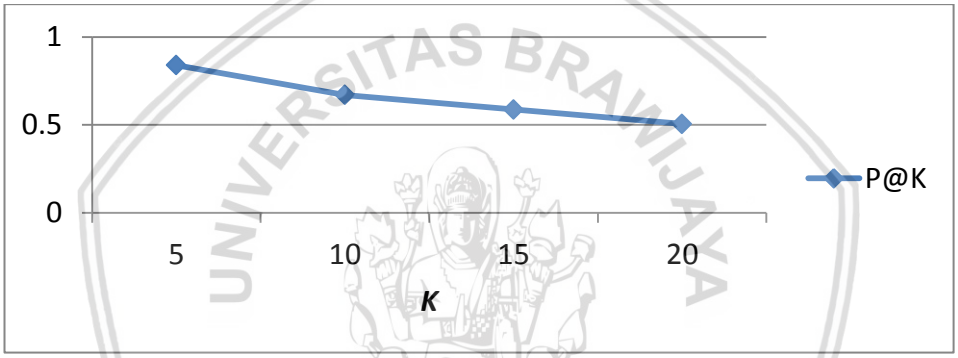
No.	ID	*Query*
1.	Q-01	*Smart home system*
2.	Q-02	*Website usability evaluation*
3.	Q-03	*User experience*
4.	Q-04	*Forecasting with neural networks*
5.	Q-05	*Wireless sensor network*
6.	Q-06	Permainan edukasi
7.	Q-07	Sistem pakar pendeteksi penyakit pada tanaman
8.	Q-08	Pengenalan wajah
9.	Q-09	Klasifikasi teks
10.	Q-10	Algoritma klastering

Penilaian relevansi antara dokumen dan *query* dilakukan secara manual terhadap 10 penilai. Masing-masing penilai memasukkan *query* dan menilai dokumen yang menurutnya relevan sampai urutan ke-5, ke-10, ke-15, dan ke-20. Hasil pengujiannya dituliskan pada **Tabel 5.2** dan kurvanya digambarkan pada Gambar 5.1. Tabel penilaian relevansi dari penguji ditampilkan secara lebih lengkap dibagian Lampiran B.

Tabel 5.2 Tabel Hasil Pengujian dengan *Precision @ K*

Query	*K*=5		*K*=10		*K*=15		*K*=20	
	Jumlah Dokumen Relevan	*P@5*	Jumlah Dokumen Relevan	*P@10*	Jumlah Dokumen Relevan	*P@15*	Jumlah Dokumen Relevan	*P@20*
Q-01	5	1	7	0,7	9	0,6	12	0,6
Q-02	5	1	9	0,9	13	0,867	15	0,75

Tabel 5.2 Tabel Hasil Pengujian dengan *Precision @ K* (lanjutan)								
Query	*K*=5		*K*=10		*K*=15		*K*=20	
	Jumlah Dokumen Relevan	*P@5*	Jumlah Dokumen Relevan	*P@10*	Jumlah Dokumen Relevan	*P@15*	Jumlah Dokumen Relevan	*P@20*
Q-03	4	0,8	8	0,8	11	0,733	13	0,65
Q-04	5	1	7	0,7	10	0,667	10	0,5
Q-05	3	0,6	8	0,8	13	0,867	18	0,9
Q-06	5	1	6	0,6	6	0,4	6	0,3
Q-07	5	1	7	0,7	7	0,467	8	0,4
Q-08	3	0,6	4	0,4	4	0,267	4	0,2
Q-09	3	0,6	5	0,5	7	0,467	7	0,35
Q-10	4	0,8	6	0,6	8	0,533	8	0,4
Rata-rata		0,84		0,67		0,5868		0,505



Gambar 5.1 Kurva Pengaruh Nilai *K* Terhadap *Precision*

Berdasarkan hasil pengujian dengan menggunakan *precision@K* dari 10 *query* didapatkan nilai presisi rata-rata mencapai 0,84, artinya, probabilitas sistem dapat menemukan kembali dokumen yang relevan dengan *query* adalah sebesar 0,84 atau 84% untuk nilai *k*=5. Nilai *precision* dipengaruhi oleh beberapa hal, antara lain nilai *top-K* yang digunakan, semakin besar nilai *K* yang digunakan, maka, nilai *precision*-nya akan semakin rendah. Selain nilai *top-K*, urutan hasil temu kembali juga berpengaruh terhadap *precision* sistem. Jika dokumen yang tidak relevan memiliki peringkat lebih tinggi, maka sistem harus melakukan proses temu kembali terhadap dokumen yang lebih banyak. Dalam pengujian kasus ini terjadi pada *Q-05* di mana nilai *K=20* memiliki nilai *precision* terbaik.

5.2 Evaluasi Perbandingan Hasil *Precision-Recall Curve*

Precision-Recall Curve (PRC) adalah sebuah kurva yang digunakan untuk menggambarkan hubungan antara *precision* dan *recall,* serta digunakan untuk menghitung nilai *precision* pada beberapa tingkat *recall.* Dengan menggunakan nilai *PRC* dapat dibandingkan nilai *precision* dan *recall* antara *query* yang sudah diterjemahkan dan *query* yang tidak diterjemahkan. Hasil evaluasi yang diambil adalah penghitungan terhadap *Q-01* sampai dengan *Q-03* sebagai sampel. Tabel hasil pengujian terhadap *Q-04* sampai dengan *Q-10* lebih lengkapnya tertera di bagian Lampiran.

Tabel 5.3 Tabel Perbandingan *Precision* dan *Recall Q-01*

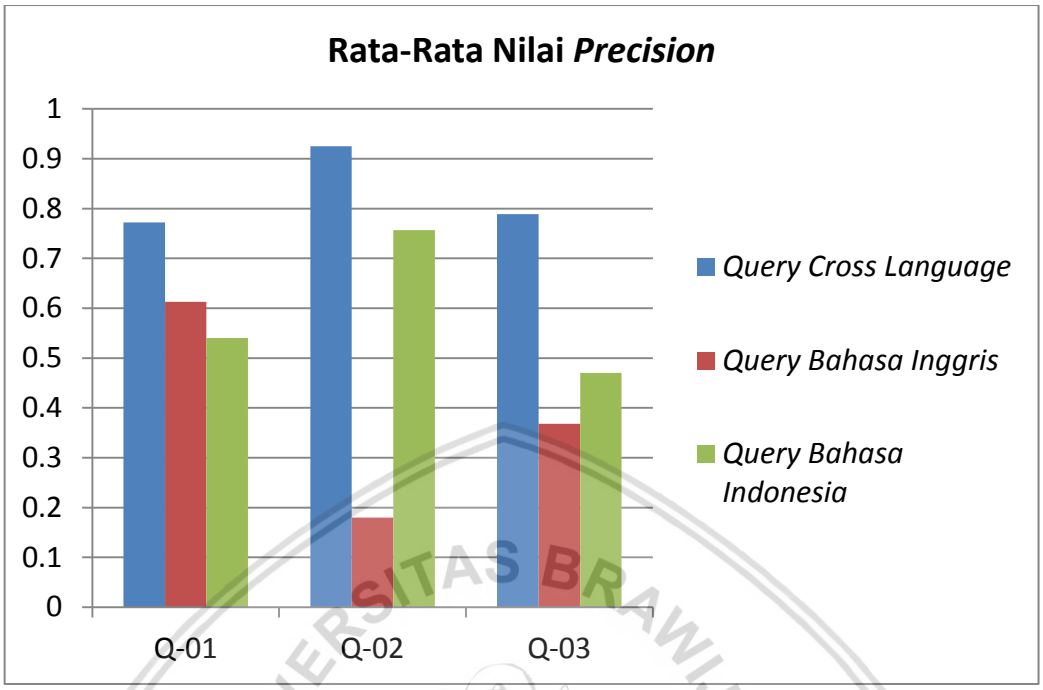
No.	Menggunakan *query* masukan dan *query* terjemahan				Hanya menggunakan *query* masukan				Hanya menggunakan *query* terjemahan			
	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*
1	D-52	Relevan	0,05	1	D-52	Relevan	0,05	1	D-792	Relevan	0,05	1
2	D-838	Relevan	0,1	1	D-644	Relevan	0,1	1	D-571	Tidak Relevan	0,05	0,5
3	D-801	Relevan	0,15	1	D-838	Relevan	0,15	1	D-570	Tidak Relevan	0,05	0,333
4	D-644	Relevan	0,2	1	D-931	Relevan	0,2	1	D-677	Tidak Relevan	0,05	0,25
5	D-928	Relevan	0,25	1	D-128	Tidak Relevan	0,2	0,8	D-763	Relevan	0,1	0,4
6	D-931	Relevan	0,3	1	D-801	Relevan	0,25	0,833	D-768	Relevan	0,15	0,5
7	D-792	Relevan	0,35	1	D-277	Tidak Relevan	0,25	0,714	D-772	Relevan	0,2	0,571
8	D-571	Tidak Relevan	0,35	0,875	D-335	Tidak Relevan	0,25	0,625	D-928	Relevan	0,25	0,625
9	D-128	Tidak Relevan	0,35	0,778	D-2	Relevan	0,3	0,667	D-927	Relevan	0,3	0,667
10	D-570	Tidak Relevan	0,35	0,7	D-389	Tidak Relevan	0,3	0,6	D-871	Relevan	0,35	0,7
11	D-677	Tidak Relevan	0,35	0,636	D-447	Tidak Relevan	0,3	0,545	D-983	Tidak Relevan	0,35	0,636
12	D-763	Relevan	0,4	0,667	D-436	Tidak Relevan	0,3	0,5	D-808	Tidak Relevan	0,35	0,583
13	D-277	Tidak Relevan	0,4	0,615	D-412	Tidak Relevan	0,3	0,462	D-530	Tidak Relevan	0,35	0,538
14	D-335	Tidak Relevan	0,4	0,571	D-131	Tidak Relevan	0,3	0,429	D-801	Relevan	0,4	0,571
15	D-768	Relevan	0,45	0,6	D-934	Tidak Relevan	0,3	0,4	D-599	Tidak Relevan	0,4	0,533
16	D-934	Tidak Relevan	0,45	0,563	D-321	Tidak Relevan	0,3	0,375	D-655	Tidak Relevan	0,4	0,5
17	D-772	Relevan	0,5	0,588	D-437	Tidak Relevan	0,3	0,353	D-904	Tidak Relevan	0,4	0,470
18	D-927	Relevan	0,55	0,611	D-572	Tidak Relevan	0,3	0,333	D-681	Relevan	0,45	0,5
19	D-871	Relevan	0,6	0,632	D-831	Tidak Relevan	0,3	0,316	D-750	Tidak Relevan	0,45	0,473
20	D-572	Tidak Relevan	0,6	0,6	D-61	Tidak Relevan	0,3	0,3	D-885	Tidak Relevan	0,45	0,45
		Rata-rata:	0,358	0,772		Rata-rata:	0,253	0,613		Rata-rata:	0,278	0,54

Tabel 5.4 Tabel Perbandingan *Precision* dan *Recall Q-02*

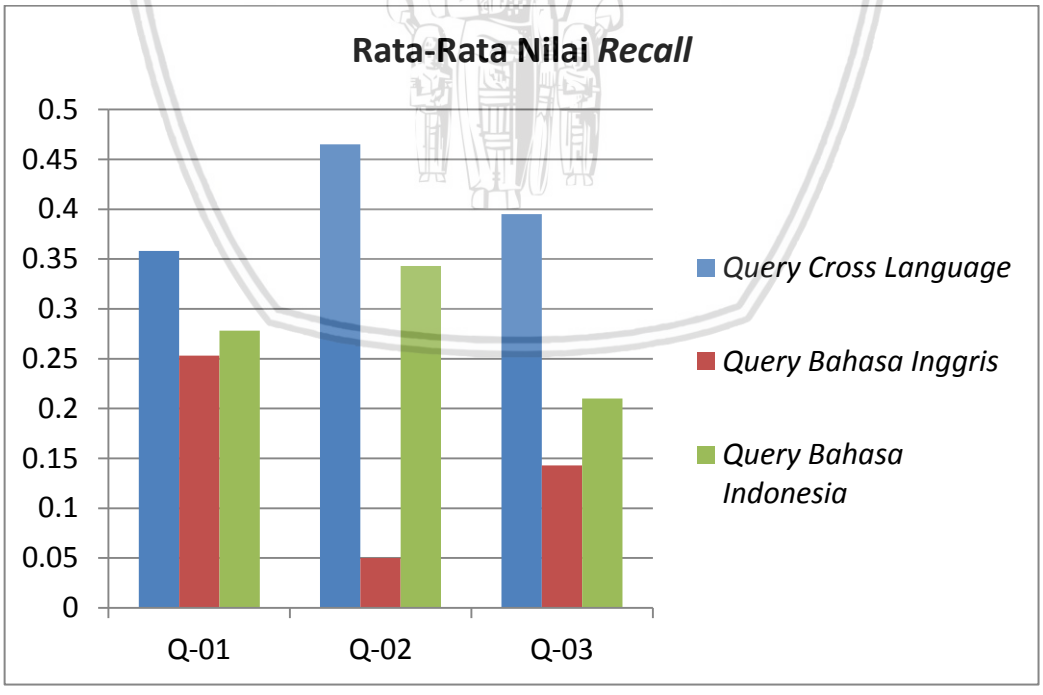
No.	Menggunakan *query* masukan dan *query* terjemahan				Hanya menggunakan *query* masukan				Hanya menggunakan *query* terjemahan			
	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*
1	D-243	Relevan	0,05	1	D-243	Relevan	0,05	1	D-700	Relevan	0,05	1
2	D-243	Relevan	0,1	1	D-203	Tidak Relevan	0,05	0,5	D-723	Relevan	0,1	1
3	D-700	Relevan	0,15	1	D-174	Tidak Relevan	0,05	0,333	D-689	Relevan	0,15	1
4	D-700	Relevan	0,2	1	D-55	Tidak Relevan	0,05	0,25	D-775	Relevan	0,2	1
5	D-723	Relevan	0,25	1	D-107	Tidak Relevan	0,05	0,2	D-625	Relevan	0,25	1
6	D-689	Relevan	0,3	1	D-225	Tidak Relevan	0,05	0,167	D-898	Relevan	0,3	1
7	D-775	Relevan	0,35	1	D-402	Tidak Relevan	0,05	0,143	D-943	Tidak Relevan	0,3	0,857
8	D-625	Relevan	0,4	1	D-183	Tidak Relevan	0,05	0,125	D-730	Relevan	0,35	0,875
9	D-898	Relevan	0,45	1	D-217	Tidak Relevan	0,05	0,111	D-936	Tidak Relevan	0,35	0,778
10	D-943	Tidak Relevan	0,45	0,9	D-36	Tidak Relevan	0,05	0,1	D-509	Relevan	0,4	0,8
11	D-723	Relevan	0,5	0,91	D-281	Tidak Relevan	0,05	0,091	D-563	Tidak Relevan	0,4	0,727
12	D-730	Relevan	0,55	0,916	D-182	Tidak Relevan	0,05	0,083	D-874	Tidak Relevan	0,4	0,667
13	D-689	Relevan	0,6	0,923	D-262	Tidak Relevan	0,05	0,077	D-607	Relevan	0,45	0,692
14	D-936	Tidak Relevan	0,6	0,857	D-228	Tidak Relevan	0,05	0,071	D-791	Tidak Relevan	0,45	0,643
15	D-775	Relevan	0,65	0,867	D-25	Tidak Relevan	0,05	0,067	D-829	Tidak Relevan	0,45	0,6
16	D-509	Relevan	0,7	0,875	D-17	Tidak Relevan	0,05	0,063	D-1000	Tidak Relevan	0,45	0,563
17	D-625	Relevan	0,75	0,882	D-491	Tidak Relevan	0,05	0,058	D-586	Tidak Relevan	0,45	0,529
18	D-203	Tidak Relevan	0,75	0,833	D-392	Tidak Relevan	0,05	0,056	D-620	Tidak Relevan	0,45	0,5
19	D-898	Tidak Relevan	0,75	0,789	D-261	Tidak Relevan	0,05	0,053	D-807	Tidak Relevan	0,45	0,474
20	D-943	Tidak Relevan	0,75	0,75	D-497	Tidak Relevan	0,05	0,05	D-963	Tidak Relevan	0,45	0,45
		Rata-rata:	0,465	0,925		Rata-rata:	0,05	0,18		Rata-rata:	0,343	0,757

Tabel 5.5 Tabel Perbandingan *Precision* dan *Recall Q-03*

No.	Menggunakan *query* masukan dan *query* terjemahan				Hanya menggunakan *query* masukan				Hanya menggunakan *query* terjemahan			
	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*	Dokumen	Relevansi	*Recall*	*Precision*
1	D-738	Relevan	0,05	1	D-457	Relevan	0,05	1	D-711	Relevan	0,05	1
2	D-711	Relevan	0,1	1	D-935	Relevan	0,1	1	D-586	Tidak Relevan	0,05	0,5
3	D-586	Tidak Relevan	0,1	0,667	D-197	Tidak Relevan	0,1	0,667	D-738	Relevan	0,1	0,667
4	D-702	Relevan	0,15	0,75	D-96	Tidak Relevan	0,1	0,5	D-702	Relevan	0,15	0,75
5	D-620	Relevan	0,2	0,8	D-427	Tidak Relevan	0,1	0,4	D-552	Tidak Relevan	0,15	0,6
6	D-970	Relevan	0,25	0,833	D-499	Tidak Relevan	0,1	0,333	D-912	Tidak Relevan	0,15	0,5
7	D-694	Relevan	0,3	0,857	D-174	Tidak Relevan	0,1	0,286	D-939	Tidak Relevan	0,15	0,429
8	D-730	Relevan	0,35	0,875	D-405	Tidak Relevan	0,1	0,25	D-710	Tidak Relevan	0,15	0,375
9	D-788	Tidak Relevan	0,35	0,778	D-159	Tidak Relevan	0,1	0,222	D-620	Relevan	0,2	0,444
10	D-457	Relevan	0,4	0,8	D-182	Tidak Relevan	0,1	0,2	D-788	Tidak Relevan	0,2	0,4
11	D-963	Relevan	0,45	0,818	D-730	Relevan	0,15	0,272	D-886	Tidak Relevan	0,2	0,363
12	D-775	Relevan	0,5	0,833	D-47	Tidak Relevan	0,15	0,25	D-694	Relevan	0,25	0,417
13	D-935	Relevan	0,55	0,846	D-963	Relevan	0,2	0,308	D-970	Relevan	0,3	0,462
14	D-552	Tidak Relevan	0,55	0,786	D-214	Tidak Relevan	0,2	0,286	D-796	Tidak Relevan	0,3	0,429
15	D-197	Tidak Relevan	0,55	0,733	D-362	Tidak Relevan	0,2	0,267	D-986	Tidak Relevan	0,3	0,4
16	D-912	Tidak Relevan	0,55	0,6875	D-954	Tidak Relevan	0,2	0,25	D-633	Tidak Relevan	0,3	0,375
17	D-943	Relevan	0,6	0,705882	D-353	Tidak Relevan	0,2	0,235	D-526	Tidak Relevan	0,3	0,353
18	D-939	Tidak Relevan	0,6	0,666667	D-283	Tidak Relevan	0,2	0,222	D-583	Tidak Relevan	0,3	0,333
19	D-705	Relevan	0,65	0,684211	D-606	Tidak Relevan	0,2	0,211	D-891	Tidak Relevan	0,3	0,316
20	D-96	Tidak Relevan	0,65	0,65	D-318	Tidak Relevan	0,2	0,2	D-547	Tidak Relevan	0,3	0,3
		Rata-rata:	0,395	0,789		Rata-rata:	0,143	0,368		Rata-rata:	0,21	0,470608



Gambar 5.2 Kurva Rata-Rata Nilai *Precision*



Gambar 5.3 Kurva Rata-Rata Nilai *Recall*

Berdasarkan data pada Tabel 5.3 sampai dengan Tabel 5.5 untuk *Q-01*, *Q-02,* dan *Q-03*, hasil temu kembali untuk 20 dokumen tertinggi menujukan bahwa dengan melakukan penerjemahan *query* dapat menaikkan nilai *precision* dan *recall*. Kurva perubahan rata-rata nilai *precision* digambarkan pada Gambar 5.2 dan perubahan rata-rata nilai *recall* digambarkan pada Gambar 5.3.

Dari total 10 *query* pencarian, 8 *query* di antaranya mengalami kenaikan nilai *precision* dan *recall*, yakni pada *Q-01, Q-02, Q-03, Q-04, Q-06, Q-08, Q-09,* dan *Q-10*. *Q-07* menujukan nilai *precision* dan *recall* yang sama antara *query* yang diterjemahkan dan tidak diterjemahkan. *Q-05* memiliki nilai *precision* dan *recall* yang lebih tinggi ketika *query* tidak diterjemahkan.

Kenaikan nilai *precision* tertinggi terjadi pada *query* yang diterjemahkan terhadap *query* berbahasa Inggris sebesar 74,5% dengan *recall* sebesar 41,5% pada pengujian *Q-02.* Perbandingan *query* yang diterjemahkan dengan *query* berbahasa Indonesia mengalami kenaikan nilai *precision* tertinggi sebesar 70,2% dengan *recall* 30,5% pada pengujian *Q-07*.

Nilai *precision* dan *recall* pada *query* yang diterjemahkan lebih tinggi karena *term* yang digunakan untuk pada pencarian lebih banyak, sehingga peringkat dokumen yang relevan juga dapat naik dan sistem dapat menemukan dokumen berbahasa Inggris dan bahasa Indonesia dengan lebih akurat.

Dalam beberapa kasus seperti pada *Q-07* yang memiliki nilai *precision* dan *recall* sama meskipun *query*-nya tidak diterjemahkan dikarenakan tidak ada dokumen yang relevan dalam pada dokumen yang berbeda bahasa (dalam kasus ini adalah dokumen berbahasa Inggris). Pada *query Q-05*, nilai *similarity* yang berpengaruh terhadap urutan dokumen dan berpengaruh juga pada nilai *precision* dan *recall*-nya.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan hasil pengujian dari implementasi sistem pencarian *cross language* yang sudah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Metode VSM dengan menerapkan *cross language* memiliki nilai akurasi sebesar 84% dalam menemukan kembali dokumen. Nilai ini berarti probabilitas sistem dapat menemukan dokumen yang relevan sebesar 0,84. Nilai ini didapatkan dari hasil pengujian dan penilaian dari 5 dokumen teratas yang ditemukan kembali. Jumlah dokumen teratas yang ditemukan kembali juga memengaruhi nilai *precision*-nya. Semakin besar nilai *top-K* dokumen yang ditemukan kembali, maka nilai *precision* sistem akan semakin rendah.

2. Penggunaan penerjemahan *query* dapat meningkatkan *precision* dan *recall* hasil temu kembali. Dari pengujian yang dilakukan terhadap 10 *query*, 8 dari 10 *query* mengalami kenaikan *precision* rata-rata sebesar 23,7% dan mencapai kenaikan *precision* tertinggi sebesar 74,5%. Kenaikan *precision* ini berarti penerapan penerjemahan *query* berhasil meningkatkan ketepatan dokumen yang ditemukan kembali dengan *query* yang dimasukan pengguna. Pada pengujian serupa, *recall* mengalami kenaikan rata-rata sebesar 12,5% dengan kenaikan *recall* tertinggi mencapai 41,5%. Kenaikan *recall* berarti meningkatkan keberhasilan sistem dalam menemukan kembali dokumen.

6.2 Saran

Saran untuk penelitian selanjutnya berdasarkan penelitian ini dari penulis adalah sebagai berikut:

1. Untuk meningkatkan akurasi hasil temu kembali dapat dilakukan dengan menambahkan jumlah dokumen untuk ditemukan kembali
2. Fitur *query expansion* dapat diterapkan untuk meningkatkan nilai *similarity* dan akurasi temu kembali
3. Dapat ditambahkan fitur-fitur lain misalnya untuk mengatasi kesalahan pengetikan *query* dari pengguna
4. Menambahkan scenario pengujian agar hasil evaluasi lebih bisa sesuai dengan permasalahan yang lebih luas

DAFTAR PUSTAKA

Babu, A. & Sindhu, L., 2014. A Survey of Information Retrieval Models for Malayalam. *International Journal of Computer Applications*, 107(14).

Hiemstra, D., 2009. Information Retrieval Models. In Goker & Davies, eds. *Information Retrieval: Searching in the 21st Century*.

Lestari, N.P., 2016. *Uji Recall And Precision Sistem Temu Kembali Informasi OPAC Perpustakaan ITS Surabaya*. Surabaya: Airlangga University.

Losee, R.M., 1998. Comparing Boolean and Probabilistic Information Retrieval Systems Across Queries and Disciplines. *J. of the American Society for Information Science*, pp.143-56.

Rusydi, I., 2014. Pemanfaatan E-Journal sebagai Media Informasi Digital. *Jurnal Iqra'*, 08(02), p.200.

Sahat, M., 2017. Studi Perbandingan Algoritma-Algoritma Stemming untuk Dokumen Teks Bahasa Indonesia. *Jurnal INKOFAR*, 1.1.

Sari, S. & Adriani, M., 2014. Learning to Rank for Determining Relevant Document in Indonesian-English Cross Language Information Retrieval using BM25. pp.309-14.

Sharma, V.K. & Mittal, N., 2016. Exploiting Parallel Sentences and Cosine Similarity for Identifying Target Language Translation. *Procedia Computer Science*, (89), pp.428-33.

Ujjwal, D., Rastogi, P. & Siddhartha, S., 2016. Analysis of Retrieval Models for Cross Language Information Retrieval.

Wu, D. & He, D., 2010. A Study of Query Translation using Google Machine System.

Yazidanyastuti, 2011. Aplikasi Analisis Halaman Website pada Mesin Pencari Google (Search Engine Google).